AD-A065 447 GENERAL RESEARCH CORP SANTA BARBARA CALIF F/G 9/2
FORTRAN AUTOMATED VERIFICATION SYSTEM (FAVS). VOLUME II. USER'S--ETC(U)
JAN 79 D M ANDREWS, R A MELTON F30602-76-C-0436

UNCLASSIFIED RADC-TR-78-268-VOL-2 NL

1 OF 2

AD
A065447

LEVEL III (12)

B.S.

AD A065447

DDC FILE COPY

RADC-TR-78-268, Volume II (of three)
Final Technical Report
January 1979

ROME AIR DEVELOPMENT CENTER

# FORTRAN AUTOMATED VERIFICATION SYSTEM (FAVS)

## User's Manual


General Research Corporation

D. M. Andrews
R. A. Melton


A065406



APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED


ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441


79 03 05 041

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-78-268, Vol II of three has been reviewed and is approved for publication.

APPROVED: Frank S LaMonica

FRANK S. LAMONICA
Project Engineer

APPROVED: Wendall C Bauman

WENDALL C. BAUMAN, Col, USAF
Chief, Information Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ISIE), Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy

1

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER<br>RADC-TR-78-268, Vol II (of three) | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER<br>Volume II. |
| 4. TITLE (and Subtitle)<br>(6) FORTRAN AUTOMATED VERIFICATION SYSTEM (FAVS). USER'S MANUAL | | 5. TYPE OF REPORT & PERIOD COVERED<br>(9) Final Technical Report, October 1976 - August 1978 |
| | | 6. PERFORMING ORG. REPORT NUMBER<br>N/A |
| 7. AUTHOR(s)<br>(10) Dorothy Rich<br>D. M. Andrews<br>D. A. Melton | | 8. CONTRACT OR GRANT NUMBER(s)<br>(15) F30602-76-C-0436 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>General Research Corporation<br>P.O. Box 6770<br>Santa Barbara CA 93111 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS<br>63701B<br>(16) 32010320 (17) 03 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>Rome Air Development Center (ISIE)<br>Griffiss AFB NY 13441 | | 12. REPORT DATE<br>(11) January 1979 |
| | | 13. NUMBER OF PAGES<br>138 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)<br>Same (12) 142p. | | 15. SECURITY CLASS. (of this report)<br>UNCLASSIFIED |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE<br>N/A |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

(18) RADC

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same (19) TR-78-268-Vol-2

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Frank S. Lamonica (ISIE)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Software
Software Testing
Software Verification
Software Documentation
FAVS
Automated Verification System

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

FAVS, for FORTRAN Automated Verification System, is a tool for analyzing source programs written in FORTRAN or DMATRAN. It is essentially a software system to be used as an aid in improving, documenting, and validating the quality of software and software testing by providing for: (1) syntax and structural analysis of the user's source program, (2) static analysis to detect inconsistencies in program structure or in the use of variables, (3) automated documentation, (4) instrumentation of the source code, (5) analysis of testing coverage, and (6) retesting guidance. A separate function that FAVS can

next page

DD FORM 1 JAN 73 1473

402 754

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

79 03 05 041

perform is the transformation of an unstructured FORTRAN program into a logically equivalent DMATRAN program.

This manual describes how to use FAVS from the beginning of the software development cycle to its completion.

FAVS has been installed on the HIS 6180 GCOS and MULTICS computer systems at the Rome Air Development Center, Griffiss AFB, and on the UNIVAC 1100/42 computer systems at the Defense Mapping Agency Aerospace Center (DMAAC) in St. Louis, MO, and the Defense Mapping Agency Topographic Center (DMATC) in Washington DC.

CONTENTS

CONTENTS (Cont.)

ILLUSTRATIONS

# 1 INTRODUCTION

FAVS (for FORTRAN Automated Verification System) is a tool to provide assistance in the various phases of software system development. It can be helpful from the very early stages of implementation, through system integration, testing, documentation and maintenance. As the software is being developed, one or more of its modules may be submitted to FAVS for a static analysis which will help detect errors or conditions which indicate the possibility of errors. The automated program documentation FAVS provides supplies a wide variety of reports that show inter- and intra-module relationships in clear, comprehensible form.

When a program is ready for testing, FAVS offers assistance before, during, and after execution. In preparation for testing, FAVS can instrument the system by automatically inserting software probes at appropriate points in the program to measure testing coverage. During an execution test these probes record information which is used to generate execution coverage analysis reports. These reports pinpoint paths in the program structure that remain to be exercised. In addition, retesting assistance is provided for generating testcases to the untested portions of the program. During the testing process, FAVS can be thought of as a partner, supplying a wide variety of automated aids to comprehensive testing activities.

A completely separate function that FAVS can perform is to transform an unstructured FORTRAN program into a structured DMATRAN program that is logically equivalent. The DMATRAN User's Guide, General Research Corporation CR-1-673/1 describes the features of this structured FORTRAN language. Figure 1.1 shows all the various capabilities of FAVS.

This manual describes how to use FAVS as an aid from the beginning to the end of the software development cycle. Information is presented in the order that the user is expected to need it. Section 2 is an overview of the type of aid FAVS provides. Section 3 explains what the

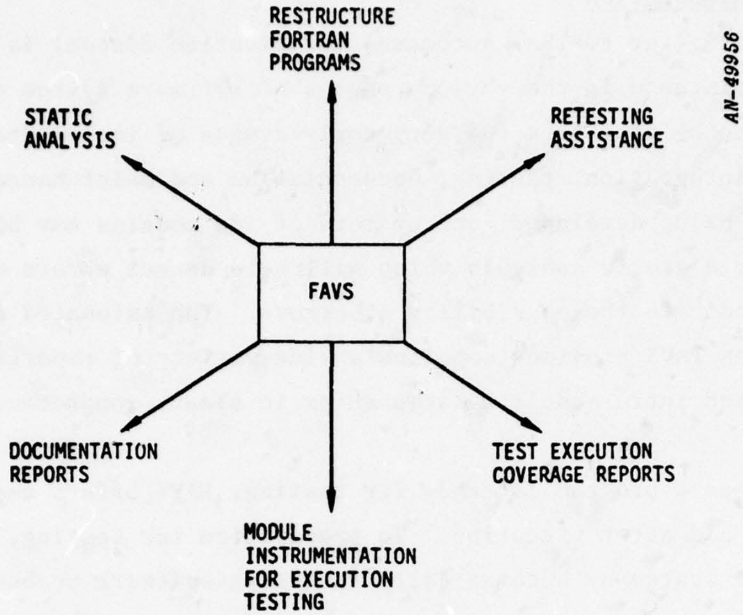



Figure 1.1. FAVS Capabilities

user has to do to use FAVS, as well as what he has to know about the details of FAVS in order to use it most effectively.

Considerable effort was expended in the design of the FAVS system to make it as easy as possible to use. The user can designate FAVS to perform a wide variety of analysis and processing by listing one or more of eight option selections. Section 4 contains a description of each FAVS option and an example of each of the reports generated by the option. Possible pitfalls due to system constraints have been categorized and itemized in Sec. 5.

When the user's program has been instrumented by FAVS and is ready for testing, a special set of ANALYZER commands are needed to generate the execution coverage analysis reports. These coverage commands are described in Sec. 6.

Although it is expected that the majority of the users of FAVS will prefer specifying the processing they want by selecting from the list of options described in Sec. 4, an introductory description of each of the FAVS segment commands (i.e., those which drive each separate function) is presented in Sec. A.1 of Appendix A for the user who would like to use individual commands. Appendix B contains a detailed description, in alphabetical order, of each segment command; sample output is included when it is generated by the command.

Appendix C contains (1) a summary of the FAVS commands, (2) a checklist for referral when using FAVS options, and (3) a summary of ANALYZER Commands. Tables listing the files used in FAVS processing at RADC and DMA installations are in Appendix D. Job streams for each installation are in Appendix E.

## 2 FAVS OVERVIEW

This section contains an overview of the way in which FAVS can aid the user not only when he is creating the code but also when he is testing and documenting it. The information presented here about what FAVS does is very general; the following sections contain more complete details of the full power of FAVS and how to use it.

Figure 2.1 shows how FAVS fits into the software development cycle to augment software analysis and testing. The additional steps are indicated by diagonal lines. The user's source code can be analyzed by FAVS and the results will be output in reports which help the user decide if the acceptance criteria are being met. FAVS can also instrument the source code prior to test execution and provide an analysis of the behavior of the program during testing.

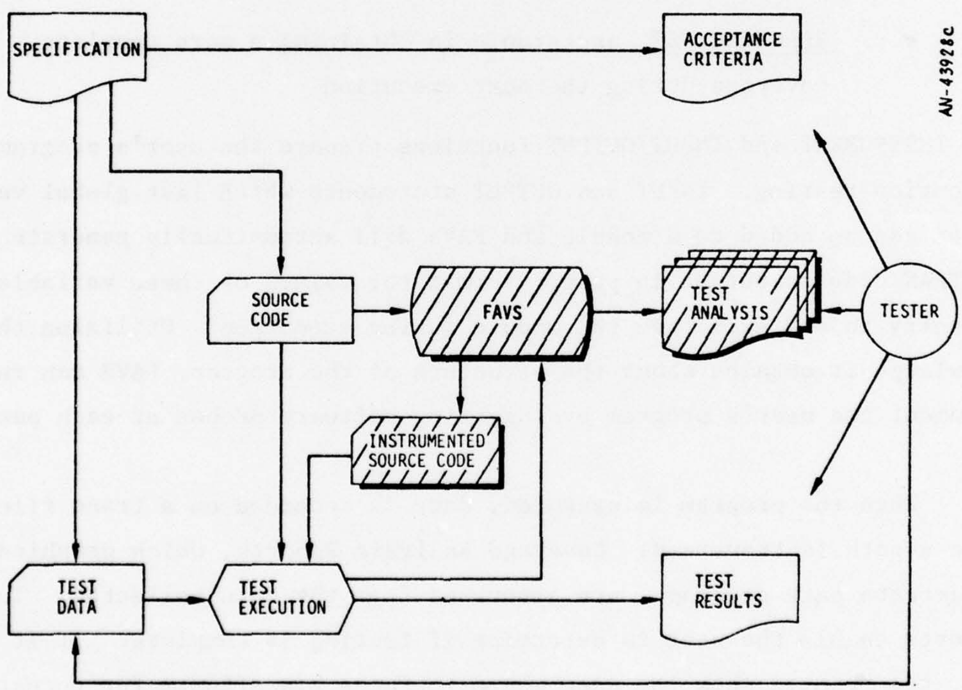



Figure 2.1. Software Analysis and Testing Augmented by FAVS

Figure 2.2 shows the sequence of source program analysis and testing when FAVS is used as a partner. If a program is written in FORTRAN, it can be restructured by FAVS into a logically equivalent DMATRAN program that will produce the same results when executed. This is a separate function that FAVS performs. The right side of the figure shows the usual sequence of events; FAVS analyzes either FORTRAN or DMATRAN source code and generates reports of the following types:

- LIST, an enhanced listing of each module
- STATIC, static analysis of each module
- DOCUMENT, interface data and relationships of modules
- SUMMARY, introductory information about modules in brief form
- INSTRUMENT, structural information about each module
- INPUT/OUTPUT, same report as INSTRUMENT
- REACHING SET, assistance in obtaining a more complete coverage during the next execution

The INSTRUMENT and INPUT/OUTPUT functions prepare the user's program for execution testing. INPUT and OUTPUT statements which list global variables can be added to a module and FAVS will automatically generate the FORTRAN code to output in proper format the values of these variables at entry to and exit from the module during execution. Utilizing the knowledge it obtains about the structure of the program, FAVS can instrument the user's program by inserting software probes at each path.

When the program is executed, data is recorded on a trace file each time a path is traversed. Coverage Analysis Reports, which graphically illustrate path coverage, are generated from the data collected. These reports enable the user to determine if testing is complete. If it is not, the reports show the user where to focus his efforts for retesting. FAVS can make further tests easier by furnishing a Reaching Set Report

SEQUENCE OF SOURCE PROGRAM ANALYSIS AND TESTING

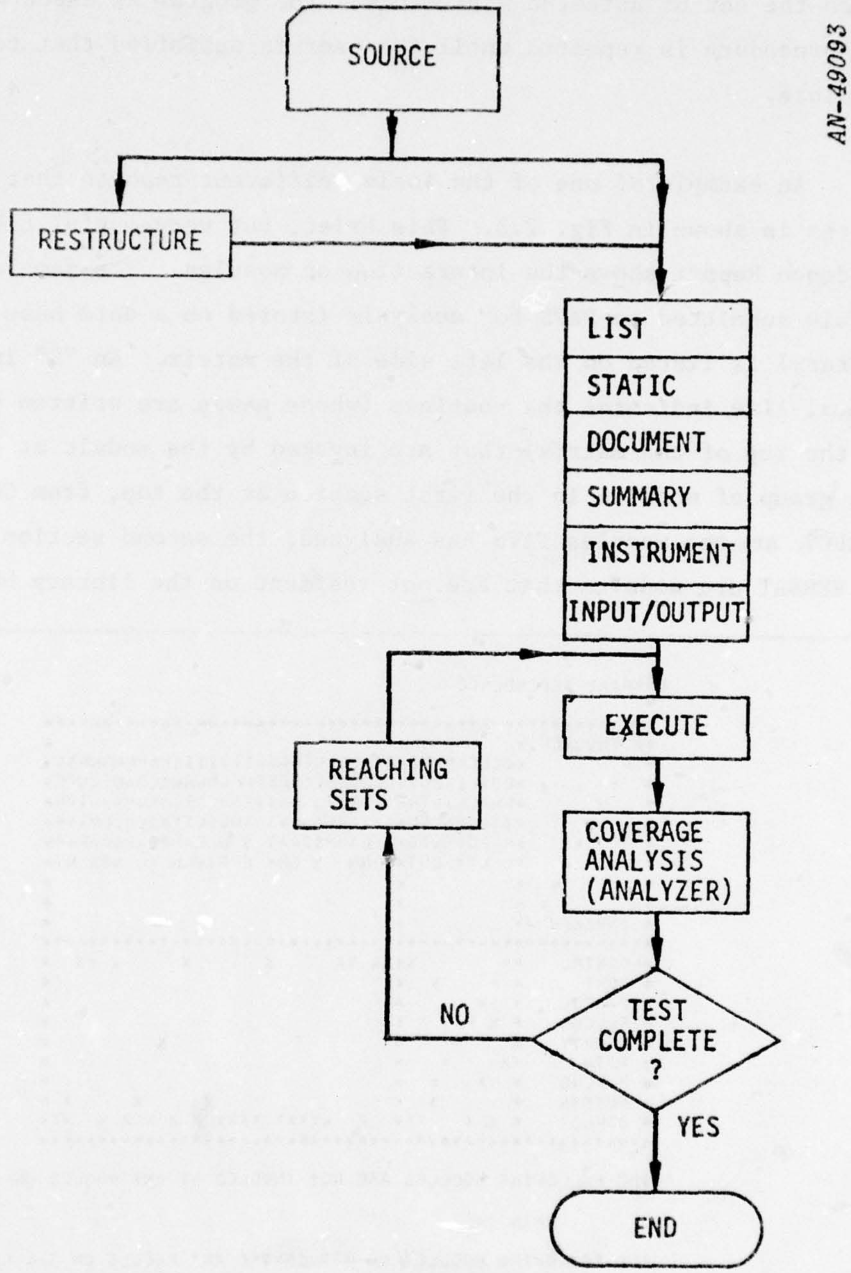



Figure 2.2. Sequence of Source Program Analysis and Testing

which lists the code from the untraversed paths. The user can then determine the values that must be assigned to the variables in order to reach the set of untested statements. The program is executed again and the procedure is repeated until the user is satisfied that testing is complete.

An example of one of the twelve different reports that FAVS generates is shown in Fig. 2.3. This brief, but very useful Library Dependence Report shows the interaction of modules. The name of each module submitted to FAVS for analysis (stored on a data base called the Library) is listed on the left side of the matrix. An "X" in the horizontal line indicates the routines (whose names are written vertically at the top of the matrix) that are invoked by the module at the left. The group of modules in the first section at the top, from CONTRL to STRUCT, are the modules FAVS has analyzed; the second section, from ACT1 to VERBAT are modules that are not resident on the library but are

---

```
LIBRARY DEPENDENCE

*******************************************************
** INVOKEE *         *                               *
* *        *CCEFKMMPS*AAABEEGGGGGGIIIIIIKKMNNNNPSV*
*  *       *OOXUEAOUT*CCSGNREEEEEOFFGNNNWCLOADEEUPE*
*   *      *NNALMIVTR*TTSSDHNNNNITCSRDCIIOAVMSW*TRR*
*    *     *TTMCPNEFL*12ICEDAGLVSOADOELTTMSEGCLPIYB*
*     *    *R POT WTC*  GARRSCAAT S UNEAHPS*BAAAF*A*
*      *   *L LNY ONT*  NN  S BRM E PTVLN 1U NBG CT*
*       *  *         *                               *
*        * *         *                               *
* INVOKER **         *                               *
*******************************************************
* CONTRL   **       X*XX XX     X       X      X XX  *
* CONT     * *    X  *                               *
* EXAMPL   *  *      *                               *
* FULCON   * X *X  X *                               *
* KEMPTY   *    *    *                  X            *
* MAIN     *X     *  *                               *
* MOVEWD   *  X   *  *                               *
* PUTFTN   *      X* *               X     X     X   *
* STRUCT   * X X  X**  X  XXXXX XXXX X X X X X   XX*
*******************************************************

THE FOLLOWING MODULES ARE NOT INVOKED BY ANY MODULE ON THE LIBRARY

        MAIN

THE FOLLOWING MODULES DO NOT INVOKE ANY MODULE ON THE LIBRARY

        EXAMPL        KEMPTY
```

Figure 2.3. Library Dependence Matrix

invoked by modules on the library. Below the matrix, MAIN is identified as the top module in the invocation hierarchy. EXAMPLE and KEMPTY are the bottom modules.

At the conclusion of each run FAVS prints a Report Index which shows the page numbers, module name, and the name of each report generated for individual modules. Some reports are an overview of all the modules; these are listed under the multi-module heading. An example of a Report Index is in Fig. 2.4.

---

```
REPORT INDEX...
                                                         PAGE        MODULE NAME

      MULTI-MODULE REPORTS

                 LIBRARY DEPENDENCE                       23
                 STATEMENT MATRIX                         24
                 COMMONS MATRIX                           25
                 READ STATEMENTS                          26
                 CROSS REFERENCE                          27
                 LIBRARY CONTENTS                         28

      SUBROUTINE EXAMPL ( INFO, LENGTH )                              EXAMPL

                 STATEMENT LISTING                         1
                 STATIC ANALYSIS                        2- 3
                 INVOCATION SPACE                          4
                 INVOCATION BANDS                          5
                 STATEMENT PROFILE                         6

      SUBROUTINE CALLER ( INFO )                                      CALLER

                 STATEMENT LISTING                         7
                 STATIC ANALYSIS                           8
                 INVOCATION SPACE                          9
                 INVOCATION BANDS                         10
                 STATEMENT PROFILE                        11

      SUBROUTINE CIRCLE ( AREA )                                      CIRCLE

                 STATEMENT LISTING                        12
                 STATIC ANALYSIS                       13- 14
                 INVOCATION SPACE                         15
                 INVOCATION BANDS                         16
                 STATEMENT PROFILE                        17

      SUBROUTINE PRNT ( AREA, RADIUS )                                PRNT

                 STATEMENT LISTING                        18
                 STATIC ANALYSIS                          19
                 INVOCATION SPACE                         20
                 INVOCATION BANDS                         21
                 STATEMENT PROFILE                        22
```

Figure 2.4. Report Index

## 3 FAVS COMMANDS

FAVS is a software system which reads as data the user's FORTRAN or DMATRAN* source text either from cards or a card image file. The type of processing to be performed on the source is specified through commands that are input to FAVS. During an initial run, a RESTART file is constructed which contains information about each module submitted for analysis. FAVS has several components which extract information from this file and produce reports. Figure 3.1 illustrates the basic elements of a FAVS analysis. The name of the files are in parentheses; see Appendix D for the logical units at RADC and DMA installations.

Before the source text to be verified is submitted to FAVS, the user should take certain preliminary steps:

1. The source text should be compiled by the UNIVAC FORTRAN V or Honeywell series 6000 FORTRAN compiler to confirm that it is free of any syntactical errors.

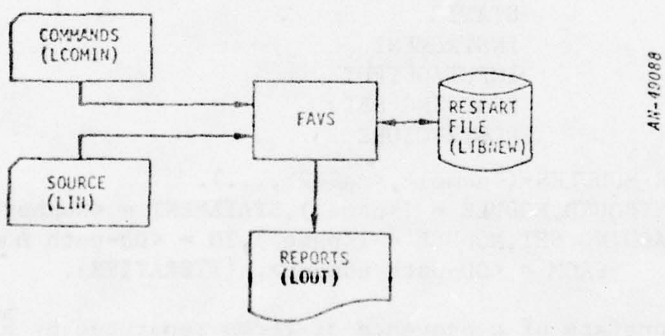



Figure 3.1. FAVS Analysis

*See DMATRAN User's Guide, General Research Corporation CR-1-673/1.

2. The program should have been previously executed if it will be dynamically tested.

FAVS processing may be specified by commands which have been developed to make FAVS easy to use. When these commands are input to FAVS,they are expanded into a set of FAVS segment commands. The emphasis in this manual is on the user commands because it is the fastest way to learn to use FAVS and, furthermore, will most likely be the way most users will prefer. Appendix A contains description and details of the segment commands for those who are interested.

The eight FAVS commands are:

```
RESTART
EXPAND
LANGUAGE=DMATRAN
FILE,PUNCH=<file name>
OPTIONS=<list>
      <list> may contain one or more of the following options,
      separated by commas:
            LIST
            DOCUMENT
            SUMMARY
            STATIC
            INSTRUMENT
            INPUT/OUTPUT
            REACHING SET
            RESTRUCTURE
FOR MODULES=(<name1>,<name2>,...).
TESTBOUND,MODULE = (<name>),STATEMENT = <number>
REACHING SET,MODULE = (<name>),TO = <DD-path number>,
      FROM = <DD-path number>, {ITERATIVE}.
```

Each command consists of a sequence of terms separated by a comma or an equal sign. These commands--one to a card--are freeform; blanks are ignored. The commands may be abbreviated by using the first four letters of the first word in the command. The names of the options also may be abbreviated the same way. The first five are the basic macro commands. TESTBOUND and REACHING SET are specification commands used with the

INSTRUMENT and REACHING SET options, respectively. The use of these two commands is included in the respective option description in Secs. 4.5 and 4.7.

## 3.1 RESTART and EXPAND

When a set of modules will be analyzed more than once, one of the commands

RESTART or EXPAND

can be used to minimize execution time and reduce costs. The first time a set of modules is processed (using any of the OPTIONS), a restart file is created on LIBNEW. This file can be saved and used in subsequent FAVS runs which further analyze the same modules (using other OPTIONS) by taking the following steps:

- On the first FAVS run, save the restart file created on LIBNEW.
- On subsequent FAVS runs, input the restart file from command. If additional modules are to be added to the restart file, use the EXPAND command.

## 3.2 LANGUAGE

If the language of the source code to be analyzed is DMATRAN, one other command is necessary.

LANGUAGE = DMATRAN.

No language specification is necessary for FORTRAN since that is the default. Whenever DMATRAN source is generated by FAVS (as a result of the RESTRUCTURE option or the INSTRUMENT option applied to DMATRAN source) it must be precompiled* before normal compilation and execution.

---

* See DMATRAN User's Guide, General Research Corporation CR-1-673/1.

## 3.3 FILE

Several of the FAVS OPTIONS (INSTRUMENT, INPUT/OUTPUT, and RESTRUCTURE) produce enhanced source output in 80-character card image form. This source normally goes to a temporary file (which may be saved after the FAVS run). If the default assignment for the source output file is not appropriate (see PUNCH in Appendix D), it may be reassigned with the command

FILE, PUNCH = <file-name>.

where <file-name> is the desired file name or file number.

## 3.4 OPTION

The command which controls the type of processing to be done by FAVS is:

OPTION(S) = <list>

The eight possible options are as follows:

- LIST - produces an enhanced source listing of each module
- SUMMARY - provides an analysis of statements, common blocks, and module dependencies.
- DOCUMENT - produces two reports for each module and a READS report, commons matrix, and an overall cross reference report for all modules.
- STATIC - produces a Static Analysis report of each module.
- INSTRUMENT - instruments the source code and writes the instrumented code to the LPUNCH file.
- INPUT/OUTPUT - same as INSTRUMENT,but also translates INPUT/OUTPUT statements into FORTRAN.
- REACHING SET - provides assistance in identifying paths to designated code segments within specified modules.

- RESTRUCTURE - generates structured DMATRAN programs from FORTRAN programs.

More than one option may be specified, only RESTRUCTURE cannot appear in conjunction with others. At least one option must be listed for any processing to take place. When there is more than one option, a comma between each is necessary. If the list exceeds 80 characters, additional OPTION commands are accepted. Continuation of the list on the next card would not be recognized. A detailed description of each option with examples of the reports the option produces may be found in Sec. 4.

## 3.5 FOR MODULES

The default is to apply the analysis requested in an OPTION command to all modules known to FAVS. Selection of specific modules for FAVS analysis is provided by the FOR MODULES command. This command has the form

FOR MODULES = (<name1>,<name2> ,....).

where <name1> and <name2> are the FORTRAN names for modules which have been input to FAVS. Main programs which do not have a program card are given the name MAIN by FAVS. The FOR MODULES command is especially useful to select specific modules on a restart file. Only one FOR MODULES command per FAVS run is allowed.

## 4 OPTION DESCRIPTIONS

This section is a reference containing a description of each option which may be selected by the user to instruct FAVS which type of processing to perform on the modules being input. An example of each type of report generated by an option follows each option description. With the exception of the RESTRUCTURE option (which is used alone), the option list may contain one or more of the remaining options in any combination. Table 4.1 shows the FAVS options and suggested uses for each.

## TABLE 4.1

## FAVS PROCESSING OPTIONS WITH SUGGESTED USES FOR EACH OPTION

| OPTIONS<br>USAGE | LIST | SUMMARY | DOCUMENT | STATIC | INSTRUMENT | INPUT/OUTPUT | REACHING SET | RESTRUCTURE |
|---|---|---|---|---|---|---|---|---|
| Software Documentation | ✓ | ✓ | ✓ | | | | | ✓ |
| Maintenance | ✓ | ✓ | ✓ | ✓ | | | | ✓ |
| Implementation | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| Obtain Interface Data | | ✓ | ✓ | ✓ | | | | |
| Trace Ranges of Variables | | | | | | ✓ | | |
| Execution Test | | | | | ✓ | ✓ | ✓ | |
| Incomplete Test Coverage | | | | | ✓ | ✓ | ✓ | |
| System Test Information | ✓ | ✓ | ✓ | | | | | |
| Single Module Information | ✓ | ✓ | | ✓ | | | | |
| Code Changes | ✓ | ✓ | ✓ | ✓ | | | | |
| Unknown Behavior | | | | ✓ | ✓ | ✓ | | |
| Integration | | | | ✓ | ✓ | | | |
| Acceptance | | ✓ | | ✓ | ✓ | | | ✓ |

4.1 LIST

The LIST option produces a source listing which shows the number of each statement, the levels of indentation, and the DD-paths. With an automatically indented listing, the programmer is relieved of having to calculate and keypunch each indentation manually; this is especially useful when changes are made to the code which would require changes in the nesting level.

An indented listing clearly indicates the control structures and makes the program much more readable, not only to the original programmer, but especially to someone unfamiliar with the code who is trying to understand it.

The indented statement listing on the output file is the sole report from the LIST option. Figure 4.1 illustrates a sample listing.

Command

OPTION = LIST

Report

Statement Listing (Fig. 4.1)

OPTION = LIST



```
STATEMENT LISTING                    SUBROUTINE EXAMPL ( INFO, LENGTH )

 NO. LEVEL    LABEL   STATEMENT TEXT...                                                    DDPATHS
--------------------------------------------------------------------------------------------------
  1                   SUBROUTINE EXAMPL ( INFO, LENGTH )                                   (  1)
  2            C
  3            C      ILLUSTRATION OF DMATRAN SYNTAX
  4            C
  5                   IF ( INFO .LE. 10 .AND. LENGTH .GT. 0 ) THEN                         (  2-  3)
  6 ( 1)              . CALL CALLER ( INFO )
  7                   ELSE
  8 ( 1)              . LENGTH = 50
  9                   ENDIF
 10                   CASEOF ( INFO + 6 )                                                  (  4-  6)
 11                   CASE ( 14 )
 12 ( 1)              . LENGTH = LENGTH - INFO
 13                   CASE ( 17 )
 14 ( 1)              . DOWHILE ( INFO .LT. 20 )                                           (  7-  8)
 15 ( 2)              . . DOUNTIL ( LENGTH .LE. INFO )
 16 ( 3)              . . . INVOKE ( COMPUTE LENGTH )
 17 ( 3)              . . . IF ( LENGTH .GE. 30 ) THEN                                     (  9- 10)
 18 ( 4)              . . . . INVOKE ( PRINT-RESULTS )
 19 ( 3)              . . . ENDIF
 20 ( 2)              . . ENDUNTIL                                                         ( 11- 12)
 21 ( 2)              . . INFO = INFO + 1
 22 ( 1)              . ENDWHILE
 23                   CASEELSE
 24 ( 1)              . DOWHILE ( LENGTH .GT. 0 )                                          ( 13- 14)
 25 ( 2)              . . INVOKE ( COMPUTE LENGTH )
 26 ( 1)              . ENDWHILE
 27                   ENDCASE
 28                   BLOCK ( PRINT-RESULTS )                                              ( 15)
 29 ( 1)              . WRITE ( 6, 1 ) INFO, LENGTH
 30 ( 1)       1      . FORMAT (10X,I5,20X,I5)
 31                   ENDBLOCK
 32                   BLOCK ( COMPUTE LENGTH )                                             ( 16)
 33 ( 1)              . LENGTH = LENGTH - 10
 34                   ENDBLOCK
 35                   RETURN
 36                   END
--------------------------------------------------------------------------------------------------
```

This report, output for each module submitted to FAVS, contains the enhanced module listing with statement numbers, nesting levels, and DD-path numbers (at procedure entry and at each conditional statement).

Figure 4.1. Statement Listing

4.2 SUMMARY

The SUMMARY option is intended to be used when a brief introduction to a set of modules is desired. It provides an analysis of statements, common blocks, and module dependencies. The statements of individual modules are classified separately as either declaration, executable, decision, or documentation. Under each classification a tabulated account of the various subtypes is listed. A separate Statement Profile report with this information, is generated for each module.

An overall view of the modules is given by the Library Dependence and the Common Matrix reports. The Dependence report shows the invokee and invoker modules and presents a picture of module dependencies. It also lists high level modules, those not invoked by any other module on the library, and low level modules which do not invoke any others on the library. The Commons Matrix report lists all the common blocks encountered in any of the modules. When program changes are made, the Dependence and Commons Matrix reports can be used to identify modules which may be affected.

A report Index from the SUMMARY option is shown in Fig. 4.2 to illustrate individual and multi-module reports.

---

```
REPORT INDEX...
                                                        PAGE        MODULE NAME

   MULTI-MODULE REPORTS

            LIBARY DEPENDENCE                            3
            COMMONS MATRIX                               4

   SUBROUTINE EXAMPL ( INFO, LENGTH )                                EXAMPL

            STATEMENT PROFILE                            1

   SUBROUTINE CALLER ( INFO )                                        CALLER

            STATEMENT PROFILE                            2
```

Figure 4.2. Report Index

The Statement Profile for Subroutine EXAMPL is shown in Fig. 4.3; (a Statement Listing of EXAMPL was used to illustrate the output from the LIST option in Fig. 4.1.). Two multi-module reports, Library Dependence and Commons Matrix, are shown in Figs. 4.4 - 4.5.

<u>Command</u>

OPTION = SUMMARY

<u>Reports</u>

| | |
|---|---|
| Statement Profile | (Fig. 4.3) |
| Library Dependence | (Fig. 4.4) |
| Common Matrix | (Fig. 4.5) |

OPTION = SUMMARY

OPTION = SUMMARY

```
STATEMENT PROFILE                 SUBROUTINE EXAMPL ( INFO, LENGTH )


     INTERFACE CHARACTERISTICS
     -------------------------

                    ARGUMENTS              2
                    ENTRY                  1
                    EXIT                   1
                    INTERNAL PROCEDURES    2
                    INVOKES                4
                    WRITE                  1

     STATEMENT      STATEMENT
     CLASSIFICATION TYPE          NUMBER        PERCENT
----------------------------------------------------------------------

     DECLARATION...

                    FORMAT          1            2.8
                    ---------------------------------
                    TOTAL           1            2.8

     EXECUTABLE...

                    ASSIGNMENT      4           11.1
                    CALL            1            2.8
                    CASE            2            5.6
                    CASEELSE        1            2.8
                    DOUNTIL         1            2.8
                    ELSE            1            2.8
                    ENDBLOCK        2            5.6
                    ENDCASE         1            2.8
                    ENDIF           2            5.6
                    ENDWHILE        2            5.6
                    END             1            2.8
                    INVOKE          3            8.3
                    RETURN          1            2.8
                    WRITE           1            2.8
                    ---------------------------------
                    TOTAL          23           63.9

     DECISION...

                    BLOCK           2            5.6
                    CASEOF          1            2.8
                    DOWHILE         2            5.6
                    ENDUNTIL        1            2.8
                    IFTRAN-IF       2            5.6
                    SUBROUTINE      1            2.8
                    ---------------------------------
                    TOTAL           9           25.0

     DOCUMENTATION...

                    COMMENT         3            8.3
                    ---------------------------------
                    TOTAL           3            8.3

----------------------------------------------------------------------
  * TOTAL PERCENTAGE MAY BE MORE THAN 100 BECAUSE OF OVERLAPPING CLASSIFICATIONS
```

This report classifies each statement of a module as either a declaration, executable, decision, or documentation statement. Under these classifications, a tabulation of the subtypes is listed.

Figure 4.3. Statement Profile

```
LIBRARY DEPENDENCE

******************************************************
** INVOKEE   *         *                             *
* *          *CCEFKMMPS*AAABEEGGGGGGIIIIIIKKMNNNNNPSV*
*  *         *OOXUEAOUT*CCSGNREEEEEOFFGNNNWCLOADEEUPE*
*   *        *NNALMIVTR*TTSSDRNNNNTTCSRDDIIOAVMSWWTRR*
*    *       *TTMCPNEFU*12ICEOAGLVSOAOOELTTMSEOCLPIYB*
*     *      *R POT WTC*  GARRSCAAT S UNEAHPSWBAAAFWA*
*      *     *L LNY DNT*  NN  S BRM E PTVLN 1U NBG CT*
*        *   *         *                             *
*          * *         *                             *
* INVOKER   **         *                             *
******************************************************
* CONTRL     **       X*XX XX      X     X     X XX  *
* CONT       * *    X  *                             *
* EXAMPL     *  *      *                             *
* FULCON     * X *X  X *                             *
* KEMPTY     *    *    *                   X         *
* MAIN       *X    *   *                             *
* MOVEWD     *  X   *  *                             *
* PUTFTN     *      X* *                X     X    X *
* STRUCT     * X X   X**   X  XXXXX XXXX X X X X X XX*
******************************************************

THE FOLLOWING MODULES ARE NOT INVOKED BY ANY MODULE ON THE LIBRARY

        MAIN

THE FOLLOWING MODULES DO NOT INVOKE ANY MODULE ON THE LIBRARY

        EXAMPL      KEMPTY
```

The interaction of all modules on the data base library is shown in the first matrix. If the library contains all modules in the user's program, this report provides a concise, complete picture of the total internal module dependencies. If the library contains a subset of the total program, this report aids in determining what modules do not interact with the component and might be better suited for another component. The modules are listed in alphabetical order.

The modules in the second matrix are not resident on the library. If the library allegedly contains all modules in the program, the external modules should consist only of system routines. If the library contains a component of the total program, this report shows the module invocation interfaces to other externals.

Considering the modules on the library as a pyramid representing the invocation hierarchy of the modules, this report also identifies the "top" and "bottom" modules in the system.

Figure 4.4. Library Dependence Matrix

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
COMMONS MATRIX

LIBRARY COMMON BLOCK MATRIX

------------------------.---------
C **         *          .        *
O * * MODULE * C C E F K.M M P S *
M *  *       * O O X U E.A O U T *
M *   *      * N N A L M.I V T R *
O *    *     * T T M C P.N E F U *
N *     *    * R   P O T.  W T C *
  *      *   * L   L N Y.  D N T *
N *       *  *          .        *
O * COMMON * *          .        *
  *         **          .        *
------------------------.---------
1  * ACCTNG  * X        .        *
2  * CARDS   * X       X.        *
3  * CONSTN  * X X     X.    X X *
4  * FORTRN  * X X      .    X X *
5  * INTERN  * X     X  .    X X *
6  * INVOKE  * X        .      X *
7  * RECNIZ  * X        .        *
8  * SESE    * X        .        *
9  * STACK   * X        .      X *
10 * STATE   * X     X  .    X X *
11 * STYPE   * X        .      X *
12 * TRACE   * X        .    X   *
13 * USEOPT  * X X     X.    X X *
14 * WARNIN  * X        .        *
------------------------.---------
```

This report lists all modules and all common blocks encountered. An "X" indicates the presence of that common in a module.

Figure 4.5. Commons Matrix

## 4.3 DOCUMENT

The DOCUMENT option generates a set of five different reports. Two are individual module reports and are produced for each module which has been input to FAVS. The other three are multi-module reports. Figures 4.6 - 4.10 contain examples and a description of each report.

Note that the Commons Matrix report (Fig. 4.10) is similar to the one produced by the SUMMARY option (Fig. 4.5), but it has considerably more information. The Commons Matrix report of the DOCUMENT option lists all the common blocks encountered and indicates, for those modules containing that common block, whether or not at least one symbol has been referenced. A second matrix shows the variables from these common blocks which are referenced by at least one module; their usage in the other modules which contain them also is itemized.

This set of reports can be used throughout the testing process. Together with the execution coverage reports, they help to identify which modules may require retesting when changes are made in the code. The Global Cross Reference report is particularly useful in finding where variables are set in order to alter test cases, and also where a variable is being used that is affected by a change in a module.

Command

OPTION = DOCUMENT

Reports

| | |
|---|---|
| Invocation Space | (Fig. 4.6) |
| Invocation Bands | (Fig. 4.7) |
| READ Statements | (Fig. 4.8) |
| Cross Reference | (Fig. 4.9) |
| Commons Matrix (Enhanced) | (Fig. 4.10) |

OPTION = DOCUMENT



```
INVOCATION SPACE                          SUBROUTINE CONT ( LABEL )
-----------------------------------------
INVOCATIONS FROM WITHIN THIS MODULE
-----------------------------------------

MODULE MOVEWD
STMT =   26          CALL MOVEWD ( 5 , 1 , LABEL , 1 , KABEL )
STMT =   28          CALL MOVEWD ( 8 , 1 , ICONT , 1 , KFTN )

INVOCATIONS TO THIS MODULE FROM WITHIN LIBRARY
-----------------------------------------

MODULE FULCON
STMT =   14          CALL CONT ( LABEL )

MODULE STRUCT
STMT =   86          CALL CONT ( LAB )
STMT =  103          CALL CONT ( LAB )
STMT =  124          CALL CONT ( LAB )
STMT =  153          CALL CONT ( LAB )
STMT =  165          CALL CONT ( LAB )
STMT =  202          CALL CONT ( LAB )
STMT =  236          CALL CONT ( LAB )
STMT =  258          CALL CONT ( LAB )
STMT =  262          CALL CONT ( LAB )
STMT =  292          CALL CONT ( LAB )
STMT =  303          CALL CONT ( LAB )
STMT =  306          CALL CONT ( LAB )
STMT =  345          CALL CONT ( NAME1 )
STMT =  361          CALL CONT ( NAME1 )
STMT =  373          CALL CONT ( NAME1 )
```

This module report shows all invocations, along with the statement numbers, to and from the specified module. It is useful in examining actual parameter usage.

Figure 4.6. Invocation Space

OPTION = DOCUMENT



```
INVOCATION BANDS              SUBROUTINE FULCON ( LABEL )
TO LEVEL 2
LEVEL     -5     -4     -3     -2     -1     0      1      2      3      4      5
-------------------------------------------------------------------------------------
                                             FULCON
                                      STRUCT        CONT
                               CONTRL                      MOVE=D
                                                    KEMPTY
                                                           MCOMP
                                                    PUTFTN
                                                           INCENT
                                                           MOVE=D
                                                           MCVE=U
                                                           SPRY=D
-------------------------------------------------------------------------------------
```

This report shows the selected module within the invocation hierarchy. At the center is the specified module. Each successive band of modules from the center to the left shows the calling modules; each successive band to the right shows the called modules. The left (calling) modules reside on the library; the right (called) modules can include modules external to the FAVS library.

Figure 4.7. Invocation Bands

OPTION = DOCUMENT



```
READ STATEMENTS


THE FOLLOWING MODULES CONTAIN READ STATEMENTS

               GETCRD
               GETINS

------------------------------------------------------------------------

READ STATEMENTS AND ASSOCIATED FORMATS
--------------------------------------

--- GETCRD ---

  16                    READ ( LUNIN, 1 ) ( LCARD ( I ), I = 1, 80 )
  17         1          FORMAT (80A1)

--- GETINS ---

  44                    READ ( 5, 1 ) ( NUN ( I ), I = 1, NOPTS )
  45         1          FORMAT (12I5 )
------------------------------------------------------------------------
```

This report provides a list of all the program modules in which a READ appears. The source statements are reproduced along with the defining FORMAT. This report may be used to locate all the points where variables are being input to the system.

Figure 4.8. READ Statements

OPTION = DOCUMENT

```
CROSS REFERENCE

GENERAL CROSS REFERENCE LISTING

MODULES INCLUDED --
          CONTRL
          CONT
          EXAMPL
          FULCON
          KEMPTY
          MAIN
          MOVEWD
          PUTFTN
          STRUCT

SYMBOL    MODULE    USED/SET/DEFINITION ( * INDICATES SET, D INDICATES DEFINITION )
ACT1      CONTRL    172
ACT2      CONTRL    174
ASSIGN    STRUCT    180
BGSCAN    CONTRL    168
CONTRL    CONTRL      1
          MAIN        2
CONT      CONT        1
          FULCON     14
          STRUCT     86  103  124  153  165  202  236  258  262  292  303  306  345
ENDER     CONTRL    183
ERROR     STRUCT     53  107  111  113  128  130  169  171  213  217  219  240  244
EXAMPL    EXAMPL      1
          MOVEWD     33
FULCON    FULCON      1
          STRUCT     84  101  122  137  160  199  234  255  275  298
GENASS    STRUCT    341
GENGO     STRUCT    369
GENLAB    STRUCT     73   81   85   98  102  123  139  141  149  152  161  164  195
                    281  283  291  299  302  305  339  340  357  360  371
GENVAR    STRUCT    179  208
GETSTM    CONTRL    164
GOTO      STRUCT     82   99  150  162  196  232  278  300  343  358
IARRY1    MOVEWD      1   23D  29*
IARRY     MOVEWD      1 . 22D  29
ICONT     CONT      24D  25D  25D  25D  25D  25D  25D  25D  25D  28
IEOF      CONTRL    29D 165  180
          KEMPTY     5D
IERROR    STRUCT     92*  93   94*  95  110  120* 121  127  158* 159  168  190* 191
                    243  253* 254  265  296* 297  309
```

This report provides a symbol cross reference listing for all modules on the library. The symbol types are variables, file names, block names, and subprogram names. Adjacent to the statement number of the symbols appearance is a flag * (or D) indicating setting or definition.

Figure 4.9. Cross Reference

OPTION = DOCUMENT OPTION = DOCUMENT

```
LIBRARY COMMON BLOCK MATRIX

--------------------------.---------
 C **          *          .        *
 O * * MODULE  * C C E F K.M M P S *
 M *  *        * O O X U E.A C U T *
 M *   *       * N N A L M.I V T R *
 O *    *      * T T M C P.N E F U *
 N *     *     * R   P O T.  W T C *
   *      *    * L   L N Y.  C N T *
 N *       *   *          .        *
 O * COMMON *  *          .        *
   *          **          .        *
--------------------------.---------
  1 * ACCTNG   * O        .        *
  2 * CARDS    * X       O.        *
  3 * CONSTN   * O X     X.    X X *
  4 * FORTRN   * O X      .    X O *
  5 * INTERN   * X     O  .    X O *
  6 * INVOKE   * O        .      X *
  7 * RECNIZ   * O        .        *
  8 * SESE     * C        .        *
  9 * STACK    * O        .      X *
 10 * STATE    * X     X  .    O X *
 11 * STYPE    * O        .      O *
 12 * TRACE    * X        .    X   *
 13 * USEOPT   * X O     O.    X X *
 14 * WARNIN   * O        .        *
-------------------------- ---------
```

```
LEGEND
--------------------------------------

COMMONS VS. MODULES
-------------------
X => AT LEAST ONE SYMBOL REFERENCED
O => NO SYMBOL EVER REFERENCED

SYMBOLS VS. MODULES
-------------------
X => SYMBOL SET AND USED
O => SYMBOL NEVER SET OR USED
S => SYMBOL SET ONLY
U => SYMBOL USED ONLY
E => SYMBOL EQUIVALENCED (OVERLAID) ONLY
A => SYMBOL IS AN ARRAY
```

```
LIBRARY COMMON SYMBOL MATRIX

--------------------------.---------
 C **          *          .        *
 O * * MODULE  * C C E F K.M M P S *
 M *  *        * O O X U E.A C U T *
 M *   *       * N N A L M.I V T R *
 O *    *      * T T M C P.N E F U *
 N *     *     * R   P O T.  W T C *
   *      *    * L   L N Y.  C N T *
 N *       *   *          .        *
 O * SYMBOL *  *          .        *
   *          **          .        *
--------------------------.---------
  2 * IECF     * U       O.        *
 13 * INDON    * O O     O.    U O *
  9 * INSTAK   * C        .      X *
 10 * ITYPE    * U     O  .    O U *
 A4 * KABEL    * C X      .    U O *
  4 * KENGTH   * O S      .    X O *
 A4 * KFTN     * O U      .    U O *
 13 * KOMFTN   * O O     O.    U U *
  5 * KSTMT    * U     O  .    U O *
A10 * LABEL    * O     X  .    O U *
  5 * LEK      * C U     U.    U U *
 10 * LENGTH   * S     O  .    C O *
 10 * LINBEG   * U     O  .    O O *
 10 * LINEND   * U     O  .    O O *
A10 * LIST     * O     O  .    O U *
 10 * LPCINT   * C     U  .    O U *
 A9 * LSTACK   * O        .      X *
 10 * LTYPE    * X     O  .    J U *
 13 * LUNFOR   * O O     O.    U O *
 13 * LUNOUT   * U O     O.    O O *
 10 * MENGTH   * C     O  .    O U *
 12 * NALTER   * C        .    U   *
 A6 * NAME1    * O        .      U *
  5 * NFATER   * U     O  .    O O *
 13 * NINDNT   * O O     O.    X O *
  5 * NLINES   * S     O  .    O O *
  6 * NOBE     * O        .      X *
 A6 * NOBLOK   * O        .      S *
 A6 * NOINV    * O        .      X *
 10 * NSTATE   * X     O  .    O O *
-------------------------- ---------
```

Two matrices are produced by this report. The first one lists all common blocks encountered in any one of the modules in the set which was analyzed. If at least one symbol was used, it is indicated with an "X". If no symbol was ever referenced in the module, this is indicated by a "0". Routines from which a common block may safely be removed are easily found.

The second matrix lists only the symbols which are used by some module; the number of the common block in which it is found is printed to the left and corresponds to the number given to the common block in the first matrix. This report is an excellent aid when changes are being made to a software system.

Figure 4.10. Commons Matrix (Enhanced)

## 4.4 STATIC

The static analysis techniques available in FAVS include:

- <u>Mode and type checking</u> which identifies possible misuse of constants and variables in expressions, assignments, and invocations.
- <u>Invocational checking</u> which validates actual invocations against formal declarations; checking for consistency in number of parameters and type.
- <u>Set and use checking</u> which uncovers possible use before set conditions and similar program abnormalities within a module.
- <u>Graph checking</u> which identifies possible errors in program control structure such as unreachable code.

A rigorous analysis of program variables, including interprocedural checking, provides FAVS with the capability to uncover subtle inconsistencies which lead to errors, such as:

- The number of parameters listed does not agree with those of the routine called.
- The mode of an actual parameter does not match that of the corresponding formal parameter.
- A parameter is listed in the calling argument list as a single, non-subscripted variable but is used in the routine as an array.
- Uninitialized variables or arrays are used.

Another consistency check is performed on the structure of the program. The graph for each module is checked to see that all statements are reachable from the module's entry and that the module's exit is reachable from each statement. Unreachable statements represent extra overhead in terms of memory space required for a module, while statements from which the exit cannot be reached represent potentially catastrophic system failures.

The output consists of a Static Analysis report for each module. A very simple program has been seeded with several errors to illustrate the type of report generated by the STATIC option; it consists of two subroutines, CIRCLE and PRNT; the Statement Listing for each, from the LIST option, is in Fig. 4.11.

The Static Analysis report for Subroutine Circle is in Fig. 4.12. It contains a Statement Analysis Summary and a Symbol Analysis Summary.

Command

OPTION = STATIC

Report

Static Analysis (Fig. 4.12)

```
STATEMENT LISTING                  SUBROUTINE CIRCLE ( AREA )                          PAGE    1

  NO. LEVEL    LABEL   STATEMENT TEXT...                                                 DDPATHS
-----------------------------------------------------------------------------------------------------
   1                   SUBROUTINE CIRCLE ( AREA )                                         (  1)
   2                   COMMON / VALUES / DIAMTR
   3                   INTEGER AREA
   4                   RADIUS = DIAMTR / 2
   5                   AREA = PI * RADIUS ** 2
   6                   IF ( AREA .GT. 50 )                                                (  2-  3)
   7 ( 1)             *.  THEN
   8                   CALL PRNT ( AREA )
   9                   ENDIF
  10                   RETURN
  11                   CALL STACK ( RADIUS, AREA )
  12                   END
-----------------------------------------------------------------------------------------------------


STATEMENT LISTING                  SUBROUTINE PRNT ( AREA, RADIUS )                    PAGE    5

  NO. LEVEL    LABEL   STATEMENT TEXT...                                                 DDPATHS
-----------------------------------------------------------------------------------------------------
   1                   SUBROUTINE PRNT ( AREA, RADIUS )                                   (  1)
   2                   PRINT 1, ( RADIUS, AREA )
   3           1       FORMAT (3X, 2(F6.2))
   4                   RETURN
   5                   END
-----------------------------------------------------------------------------------------------------
```

These Statement listings were generated by the LIST option.

Figure 4.11. Statement Listings

```
STATIC ANALYSIS                SUBROUTINE CIRCLE ( AREA )
 SEQ NEST SOURCE                                                                  UNKNOWN EXTERNALS
---------------------------------------------------------------------------------------------------
   1              SUBROUTINE CIRCLE ( AREA )
   2              COMMON / VALUES / DIAMTR
   3              INTEGER AREA
   4              RADIUS = DIAMTR / 2
   5              AREA = PI * RADIUS ** 2
                  ---------------------------------------------------------------
                  -                      MODE WARNING                           -
                  -  LEFT HAND SIDE HAS MODE INTEGERRIGHT HAND SIDE HAS MODE REAL -
                  ---------------------------------------------------------------
   6              IF ( AREA .GT. 50 ) THEN
   7 ( 1)         .  CALL PRNT ( AREA )
                  ---------------------------------------------------------------
                  -                       CALL ERROR                            -
                  -       PRNT       CALLED WITH  1 ACTUALLY HAS  2 ARGUMENTS   -
                  ---------------------------------------------------------------
                  ---------------------------------------------------------------
                  -                       CALL ERROR                            -
                  -PARAMETER   1 OF PRNT      .ACTUAL PARAMETER HAS MODE INTEGER -
                  -                           .FORMAL PARAMETER HAS MODE REAL    -
                  ---------------------------------------------------------------
   8              ENDIF
   9              RETURN
  10              CALL STACK ( RADIUS, AREA )
                  ---------------------------------------------------------------
                  -                      GRAPH WARNING                          -
                  -       STATEMENT  10  IS UNREACHABLE OR IS IN AN INFINITE LOOP -
                  ---------------------------------------------------------------
                                                                                  STACK
  11              END
-------------------------------------------------------------------------------
                   STATEMENT ANALYSIS SUMMARY          ERRORS  WARNINGS
                   --------------------------          ------  --------
                   GRAPH CHECKING                         0        1
                   CALL CHECKING                          2        6
                   MODE CHECKING                          0        1

                                  1ST  TOTAL  LAST   IN/OUT ACTUAL PHYSICAL
 NAME      SCOPE      MODE        STMT  USES  STMT     USE   USE    UNITS
-------------------------------------------------------------------------------

 AREA      PARAMETER INTEGER       1     6     10           BOTH

 DIAMTR    VALUES    REAL          2     2      4           INPUT

 RADIUS    LOCAL     REAL          4     3     10

 PI        LOCAL     REAL          5     1      5
                   ---------------------------------------------------------------
                   -                       SET/USE WARNING                       -
                   -  VARIABLE PI          MAY BE USED BEFORE BEING ASSIGNED A VALUE -
                   ---------------------------------------------------------------
-------------------------------------------------------------------------------
                   SYMBOL ANALYSIS SUMMARY             ERRORS  WARNINGS
                   -----------------------             ------  --------
                   SET/USE CHECKING                       0        1
```

The Statement Analysis Summary contains the warning and error messages interspersed appropriately in the code. Unknown externals, routines called which are not in the set submitted to FAVS, are listed on the right side of the printout. A tabulation of the errors and warnings is listed at the bottom.

The Symbol Analysis Summary shows the name, scope, and mode of each symbol in any executable statement in the module. The actual use of global variables is defined as INPUT, OUTPUT, or BOTH. For any variable that is used before being assigned a value or set and not used, a warning indicates the condition which could lead to errors.

Figure 4.12. Static Analysis

## 4.5 INSTRUMENT

Figure 4.13 illustrates FAVS instrumentation of a FORTRAN or DMATRAN program to prepare it for an execution coverage test. The command

OPTION = INSTRUMENT

causes the set of input modules to be instrumented. The instrumented modules will be written to file LPUNCH inlaphabetical order. A DD-path Definitions Report will be generated for each instrumented module.

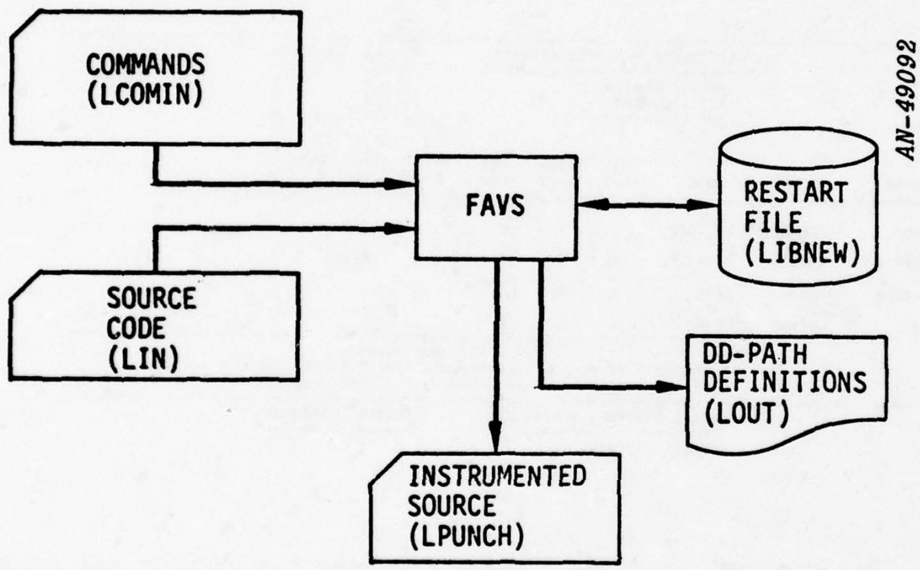



Figure 4.13. FAVS Instrumentation

A DD-path is a sequence of executable statements emanating from a conditional statement and continuing to the next conditional statement. Since complete DD-path testing means exercising all possible outways of conditional statements, this is a more rigorous testing measure than exercising all program statements. All of FAVS execution coverage reports are presented in terms of DD-path, not statement, coverage.

INSTRUMENT inserts a set of probe statements into each module. The probe statements are inserted into the source text at each entry and each exit of the modules and at each statement which begins a DD-path. Each probe includes a call to a data collection routine which records information concerning the flow of control in the executing module(s). A special probe is inserted at the end of the main program to signal the end of test execution. The user can also have this special probe inserted at other points in his code, which has the effect of breaking one test execution into multiple test cases.

The instrumented source text is written to file LPUNCH, either in DMATRAN or FORTRAN depending on the language being processed. The file can be input to the FORTRAN compiler (after first being processed by the DMATRAN precompiler if that is the source language). The instrumented object code is then ready for loading and test execution along with a FAVS supplied data collection routine.

During execution of the instrumented program, the probes record on the LTEST file a summary of execution data which resulted from processing the set of test cases input for this run.

There is a special instrumentation command which allows the user to insert special probes into his instrumented code which delineate test cases within the test execution. The user specifies a statement within a given module. Before each execution of this statement, the last test case is terminated and a new test case is begun. The form of the command for identifying a test execution boundary is:

TESTBOUND,MODULE = (<name>),STATEMENT = <number>

where <number> is the FAVS statement number in module <name> where the test-case delineation probe is desired. The probe is inserted before the number specified; therefore, the number should be that of the first statement not to be included in the test case. Up to ten TESTBOUNDS may be specified during any one instrumented run. All must immediately follow the OPTIONS command (preceding all REACHING SET commands).

The instrumented code is written on LPUNCH. The output of this step is a DD-path Definitions report, as shown in Fig. 4.14. It is an indented source listing of an individual module with additional DD-path information. At each decision point, the DD-path generated is described in terms of its decision outways. When measuring testing coverage, the user can refer to this report to associate the DD-path definitions with his original source text.

**Commands**

OPTION = INSTRUMENT
TESTBOUND,MODULE = (<name>),STATEMENT = <number>

**Report**

DD-path Definitions (Fig. 4.14)

```
DD-PATH DEFINITIONS               SUBROUTINE EXAMPL ( INFO, LENGTH )

-----------------------------------------------------------------------------------------------
  1               SUBROUTINE EXAMPL ( INFO, LENGTH )
                                                              ** DDPATH   1 IS PROCEDURE ENTRY
  2          C
  3          C    ILLUSTRATION OF DMATRAN SYNTAX
  4          C
  5               IF ( INFO .LE. 10 .AND. LENGTH .GT. 0 ) THEN
                                                              ** DDPATH   2 IS TRUE BRANCH
                                                              ** DDPATH   3 IS FALSE BRANCH
  6 ( 1)          .  CALL CALLER ( INFO )
  7               ELSE
  8 ( 1)          .  LENGTH = 50
  9               ENDIF
 10               CASEOF ( INFO + 6 )
                                                              ** DDPATH   4 IS BRANCH OUTWAY  1
                                                              ** DDPATH   5 IS BRANCH OUTWAY  2
                                                              ** DDPATH   6 IS BRANCH OUTWAY  3
 11               CASE ( 14 )
 12 ( 1)          .  LENGTH = LENGTH - INFO
 13               CASE ( 17 )
 14 ( 1)          .  DOWHILE ( INFO .LT. 20 )
                                                              ** DDPATH   7 IS LOOP AGAIN
                                                              ** DDPATH   8 IS LOOP ESCAPE
 15 ( 2)          .  .  DOUNTIL ( LENGTH .LE. INFO )
 16 ( 3)          .  .  .  INVOKE ( COMPUTE LENGTH )
 17 ( 3)          .  .  .  IF ( LENGTH .GE. 30 ) THEN
                                                              ** DDPATH   9 IS TRUE BRANCH
                                                              ** DDPATH  10 IS FALSE BRANCH
 18 ( 4)          .  .  .  .  INVOKE ( PRINT-RESULTS )
 19 ( 3)          .  .  .  ENDIF
 20 ( 2)          .  .  ENDUNTIL
                                                              ** DDPATH  11 IS LOOP ESCAPE
                                                              ** DDPATH  12 IS LOOP AGAIN
 21 ( 2)          .  .  INFO = INFO + 1
 22 ( 1)          .  ENDWHILE
 23               CASEELSE
 24 ( 1)          .  DOWHILE ( LENGTH .GT. 0 )
                                                              ** DDPATH  13 IS LOOP AGAIN
                                                              ** DDPATH  14 IS LOOP ESCAPE
 25 ( 2)          .  .  INVOKE ( COMPUTE LENGTH )
 26 ( 1)          .  ENDWHILE
 27               ENDCASE
 28               BLOCK ( PRINT-RESULTS )
                                                              ** DDPATH  15 IS A PROCEDURE ENTRY
 29 ( 1)          .  WRITE ( 6, 1 ) INFO, LENGTH
 30 ( 1)     1    .  FORMAT (10X,I5,20X,I5)
 31               ENDBLOCK
 32               BLOCK ( COMPUTE LENGTH )
                                                              ** DDPATH  16 IS A PROCEDURE ENTRY
 33 ( 1)          .  LENGTH = LENGTH - 10
 34               ENDBLOCK
 35               RETURN
 36               END
-----------------------------------------------------------------------------------------------
```

This report is useful for testing purposes because it defines the outways of all decisions and makes the decision points more visible by omitting the intervening sequential statements.

Figure 4.14. DD-Path Definitions

## 4.6 INPUT/OUTPUT

Additional information may be gathered during the execution test by inserting INPUT and OUTPUT statements into each source module. The INPUT statements list the global variables (either parameters or in common) that will have a value whenever the routine is invoked; the OUTPUT statements list variables that will be assigned a value in the routine. An INPUT variable may also be an OUTPUT variable. The INPUT/OUTPUT option provides a dynamic tracing of the values of the program variables by translating the INPUT and OUTPUT statements into FORTRAN code.

A type specification must be provided for each variable so the value will be printed with the correct format. Any variable whose type is not listed will not be printed. The syntax to provide type information is:

```
INPUT (/<type>/<variable list>,/<type>/<variable list>,...)
OUTPUT (/<type>/<variable list>,/<type>/<variable list>,...)
```

<type> may be REAL, INTEGER, HOLLERITH, or LOGICAL or the respective abbreviations for each, R , I , H , or L .

<variable list> may contain non-subscripted variable names, array names, individual elements of an array, or an array subrange, such as (LIMIT(I), I = M,N) where LIMIT is an array with a dimension of at least N and I is a variable whose value will be undefined after the INPUT or OUTPUT statement is executed.

Some specific examples are:

```
      INPUT (/I/NUMBER,(LIMIT(I),I=M,N),/R/AREA,RANGE,
     *        /L/DEBUG,/H/TEST)
      OUTPUT(/REAL/AREA,/LOGICAL/DEBUG)
```

The INPUT and OUTPUT statements are turned into comments by this option of FAVS, so they may be left in the code when the instrumented code is compiled.

The INPUT/OUTPUT option also performs the same functions as the INSTRUMENT option, so the instrumented code on LPUNCH may be used in the same way as described in Sec. 4.5.

The output of this option is the inclusion of the FORTRAN or DMATRAN translation of the INPUT and OUTPUT statements in the code written on LPUNCH. When the program is executed, the entry and exit values of the variables with type specifications listed in INPUT and OUTPUT assertions, will be reported. In addition, a DD-path Definitions report identical to the one from the INSTRUMENT option will be generated.

<u>Command</u>

OPTION = INPUT/OUTPUT

<u>Report</u>

DD-Path Definitions (Fig. 4.14)

## 4.7 REACHING SET

The analysis specified by the REACHING SET option executes the module retesting assistance of FAVS. Presuming that a set of untested DD-paths has been isolated, the user can identify a section of code he desires to exercise. He inputs the desired DD-path number to be "reached," and FAVS generates the reaching set of paths from module entry or from a designated DD-path up to the second DD-path number which has been specified. The user may specify either iterative (explained below) or non-iterative reaching sets to be generated. FAVS prints a list of DD-paths on the reaching set. With this output, the user is able to identify which parts of the program need to be executed (and therefore which program values need to be modified) for the selected DD-path to be executed. Once this determination is made, test cases can be constructed, and the user may rerun Test Execution to ascertain the additional program coverage provided by the new set of test cases.

The command

```
OPTION = REACHING SET
```

enables reaching set analysis to be performed. However, no analysis is performed unless one or more reaching sets are specified. The command for specifying a reaching set is:

```
REACHING SET,MODULE= (<name>),TO= <DD-path number>,
        FROM= <DD-path number>,{ITERATIVE}.
```

The above command generates a non-iterative reaching set. The reaching set which includes all possible iterative paths may be generated by appending ITERATIVE (preceded by a comma) to this command.

A Reaching Set report is in Fig. 4.15; it lists the set of DD-paths within the reaching set, followed by the source statements which make up that set of paths.

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC



OPTION = REACHING SET

```
REACHING SET ANALYSIS                    SUBROUTINE CLASS ( K, ITYP )

               NON-ITERATIVE REACHING SET FROM DD-PATH   8  TO  DD-PATH  40
               DDPATHS IN REACHING SET
                              8     9    10    11    12    13    14    15    17    18    22
                       23    24    25    26    27    28    36    39    40    43    44    47

               SOURCE CODE IN REACHING SET
----------------------------------------------------------------------
                              . . .
   73 ( 1)                    .  JCH = K ( J )
   74 ( 1)                    .  IF ( JCH .EQ. KBLNK )                  (  8-  9)
   75 ( 2)                    .  .  GOTO 26
   76 ( 1)                    .  IF ( JHOLL ) 12, 12, 7                 ( 10- 12)
   77 ( 1)        7           .  DO 8 L = 1, 10
   78 ( 2)                    .  .  IF ( JCH .EQ. KDEC ( L ) )          ( 13- 14)
   79 ( 3)                    .  .  .  GOTO 10
   80 ( 1)        8           .  CONTINUE                               ( 15- 16)
   81 ( 1)                    .  IF ( JHOLL - 1 ) 11, 11, 9             ( 17- 19)
                              . . .
   84 ( 1)       10           .  JHOLL = JHOLL + 1
   85 ( 1)                    .  GOTO 25
   86 ( 1)       11           .  JHOLL = 0
   87 ( 1)       12           .  IF ( JCH .EQ. KLPAR )                  ( 22- 23)
   88 ( 2)                    .  .  GOTO 20
   89 ( 1)                    .  IF ( JCH .EQ. KRPAR )                  ( 24- 25)
   90 ( 2)                    .  .  GOTO 18
   91 ( 1)                    .  IF ( JCH .EQ. KCMA )                   ( 26- 27)
   92 ( 2)                    .  .  GOTO 22
   93 ( 1)                    .  IF ( JCH .EQ. KEQ )                    ( 28- 29)
   94 ( 2)                    .  .  GOTO 23
                              . . .
  100 ( 1)       18           .  JSW = JSW - 1
  101 ( 1)                    .  IF ( JSW ) 19, 19, 25                  ( 34- 36)
                              . . .
  104 ( 1)       20           .  JSW = JSW + 1
  105 ( 1)       21           .  JHOLL = 1
  106 ( 1)                    .  GOTO 25
  107 ( 1)       22           .  IF ( JSW ) 30, 30, 21                  ( 37- 39)
  108 ( 1)       23           .  IF ( JSW ) 24, 24, 32                  ( 40- 42)
  109 ( 1)       24           .  JEQ = 1
  110 ( 1)       25           .  IF ( ISW ) 26, 26, 27                  ( 43- 45)
  111            26           CONTINUE                                  ( 46- 47)
                              . . .
----------------------------------------------------------------------
```

This report shows which DD-paths must be traversed, beginning with a specified DD-path to reach the target DD-path. Both the beginning and the ending DD-path numbers are designated by the user in the REACHING SET specification command. Coordination of this report with DD-Path Definitions report allows the user to determine what values must be supplied to the variables to affect the decision predicates so the appropriate path will be taken.

Figure 4.15. Reaching Set

## 4.8 RESTRUCTURE

The third FAVS capability is demonstrated in Fig. 4.16. The RESTRUCTURE option translates existing FORTRAN programs into the structured language, DMATRAN. FAVS reads a FORTRAN source text, creates a data base, provides an indented listing of the FORTRAN source on the output file, and writes the structured DMATRAN modules on a file. To obtain a listing, the DMATRAN file is input to the DMATRAN preprocessor. Refer to the DMATRAN User's Guide, General Research Corporation CR-1-673/1.

---

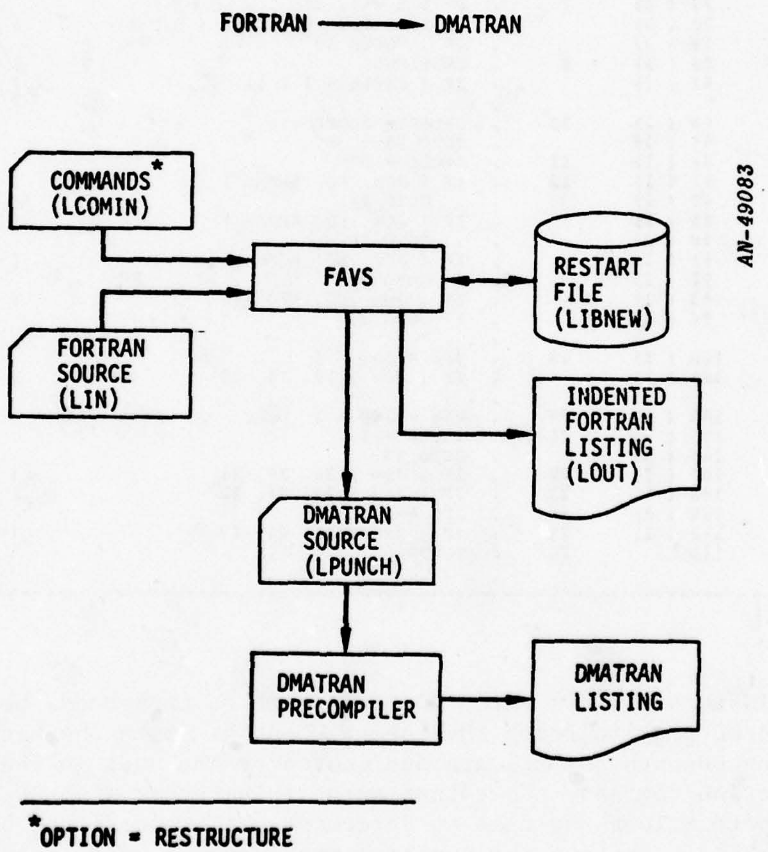



Figure 4.16. From FORTRAN to DMATRAN

Structuring does not change the logic of the original program; instead it reveals the structure of the algorithm so that it may be more readily understood. The RESTRUCTURE option is useful when existing FORTRAN programs are going to be maintained, modified, documented, or studied. The structuring process is performed once, and the resultant program can be listed and executed using the DMATRAN preprocessor.

FAVS replaces FORTRAN control statements with the following DMATRAN statement constructs:

- The IF...THEN...ELSE...END IF construct to provide block structuring of conditionally executable sequences of statements.
- The DO WHILE...END WHILE construct to permit iteration of a code segment while a specified condition remains true.
- The DO UNTIL...END UNTIL construct to permit iteration until a specified condition becomes true.

Structured programs often use the same code more than once. FAVS has the capability to isolate such segments of code and incorporate them into a BLOCK construct and add INVOKE statements in appropriate places. To make a program more readable, sections of code containing more than 100 lines are also put into a BLOCK construct and replaced with an INVOKE statement.

If the RESTRUCTURE option is selected, no other options will be processed.

The only input required by FAVS is a FORTRAN program in card image form that is compilable. More than one routine may be submitted at the same time. Note that the statement labels in the range 10000 to 19999 may be duplicated when the restructured DMATRAN source is precompiled by the DMATRAN precompiler.

The output will consist of a Statement Listing for each FORTRAN module, as shown in the example in Fig. 4.17 for Subroutine BSORT. The DMATRAN modules are written on LPUNCH in card image form. The file on LPUNCH may then be put through the DMATRAN preprocessor to obtain the indented listing of the restructured module. The DMATRAN version of the Subroutine BSORT is shown in Fig. 4.18.

Command

OPTION = RESTRUCTURE

Report

Statement Listing (Fig. 4.17)

```
STATEMENT LISTING                         SUBROUTINE BSORT ( N, ARRAY )

  NO. LEVEL   LABEL   STATEMENT TEXT...
--------------------------------------------------------------------------
   1                   SUBROUTINE BSORT ( N, ARRAY )
   2                   DIMENSION ARRAY ( 100 )
   3                   DO 1 I = 2, N
   4 ( 1)              .  IF ( ARRAY ( I - 1 ) .LE. ARRAY ( I ) )
   5 ( 2)             *.  .  GOTO 1
   6 ( 1)              .  SMALL = ARRAY ( I )
   7 ( 1)              .  ARRAY ( I ) = ARRAY ( I - 1 )
   8 ( 1)              .  J = I - 2
   9 ( 1)     2        .  IF ( J .LT. 1 )
  10 ( 2)             *.  .  GOTO 4
  11 ( 1)              .  IF ( SMALL .LT. ARRAY ( J ) )
  12 ( 2)             *.  .  GOTO 3
  13 ( 1)     4        .  ARRAY ( J + 1 ) = SMALL
  14 ( 1)              .  GOTO 1
  15 ( 1)     3        .  ARRAY ( J + 1 ) = ARRAY ( J )
  16 ( 1)              .  J = J - 1
  17 ( 1)              .  GOTO 2
  18          1        CONTINUE
  19                   RETURN
  20                   END
--------------------------------------------------------------------------
```

**This report is a source** listing of the original FORTRAN module.
**It is enhanced by indentation and** statement and nesting level numbers.

**Figure** 4.17. Statement Listing in FORTRAN

```
SEQ NEST SOURCE


           SUBROUTINE BSORT ( N, ARRAY )
           DIMENSION ARRAY ( 100 )
           I = 2
           DO UNTIL ( I .GT. N )
   1       .    IF ( ARRAY ( I - 1 ) .LE. ARRAY ( I ) ) THEN
   2       .    .    I = I + 1
   1       .    ELSE
   2       .    .    SMALL = ARRAY ( I )
   2       .    .    ARRAY ( I ) = ARRAY ( I - 1 )
   2       .    .    J = I - 2
   2       .    .    NEXIT = 0
   2       .    .    DO WHILE ( NEXIT .EQ. 0 )
   3       .    .    .    IF ( J .GE. 1 ) THEN
   4       .    .    .    .    IF ( SMALL .LT. ARRAY ( J ) ) THEN
   5       .    .    .    .    .    ARRAY ( J + 1 ) = ARRAY ( J )
   5       .    .    .    .    .    J = J - 1
   4       .    .    .    .    ELSE
   5       .    .    .    .    .    NEXIT = 2
   4       .    .    .    .    ENDIF
   3       .    .    .    ELSE
   4       .    .    .    .    NEXIT = 1
   3       .    .    .    ENDIF
   2       .    .    ENDWHILE
   2       .    .    ARRAY ( J + 1 ) = SMALL
   2       .    .    I = I + 1
   1       .    ENDIF
           ENDUNTIL
           RETURN
           END
```

This is not a report by FAVS, although it is the result of the RESTRUCTURE option of the FAVS analysis. The DMATRAN listing is obtained by using the file on LPUNCH as input to the DMATRAN preprocessor.

Figure 4.18. Restructured Module in DMATRAN

# 5 FAVS CONSTRAINTS

FAVS imposes certain restrictions on the size of the restart file, the command language, and the source text to be analyzed. Most of the limitations based on size are generous (e.g., the maximum number of nested IF statements is one hundred).

FAVS is capable of handling quite large source text files. Unusually large programs may have to be processed by several successive executions, each operating on a separate file of modules.

Universal and syntax constraints (affecting all of FAVS processing) are listed first. The remaining constraints are listed in sections according to the option they affect.

## 5.1 UNIVERSAL CONSTRAINTS

- Maximum of one card for any given command
- Maximum of 24 commas in any given command
- Maximum of 50 data base tables during any execution.
- Maximum of 250 separately compilable modules may be analyzed at one time (i.e., total modules on RESTART file).
- Maximum of 80 characters per source card image read.
- The maximum number of DD-paths which can begin at a statement is 50.
- The maximum number of statements on a single DD-path is 100.
- The sizes of the two random files LIBNEW and LIBWSP are established using a DEFINE FILE statement in the MAIN routine. The current sizes are 500 records (of 500 words each).

## 5.2 SYNTAX CONSTRAINTS

The following implementation constraints are the current ones which must be observed:

- Each module placed on the same library just have a unique name. The first six characters should be unique.
- If any errors are detected in the source, one or more statements on the RESTART file may be flagged as not parsed.
- Maximum of 100 DO statements in FORTRAN program.
- Maximum nesting depth of 25 DOs in FORTRAN.
- Maximum of 19 ENTRY statements in FORTRAN.
- Comments may not appear within statements.
- DELETE, START EDIT, STOP EDIT are not recognized.
- Switch labels may appear only in assigned GO-TO statements.
- ** is the only valid exponentiation symbol.
- No parameter list may have more than 20 parameters.

## 5.3 DOCUMENT CONSTRAINTS

- Maximum bandwidth of five specified in BAND analysis.
- Only the first 100 modules on the restart file are processed.

## 5.4 SUMMARY CONSTRAINTS

- Only the first 100 modules on the RESTART file are processed.

## 5.5 INSTRUMENT CONSTRAINTS

- A maximum of five testbounds may be specified.
- No FORTRAN labels between a range of 7777 and 8999.
- No routines named SPROB1 and SPROB2.
- The maximum number of DD-paths in one module is 9999.

## 5.6 REACHING SET CONSTRAINTS

- Analysis is limited to modules with less than 1600 DD-paths and less than 3200 statements.

## 5.7 RESTRUCTURE CONSTRAINTS

- The RESTRUCTURE option is used alone.

# 6 ANALYZER COMMANDS

A variety of coverage analysis reports can be generated from data collected during execution of a program containing one or more modules that have been instrumented by FAVS. (The INSTRUMENT option was discussed in Sec. 4.5.) Figure 6.1 shows the execution coverage sequence beginning with FAVS instrumentation of a program, through the usual compilation and execution (shown inside dashes), to the input of ANALYZER commands which then generate coverage reports; the entire sequence can be performed in the same run.

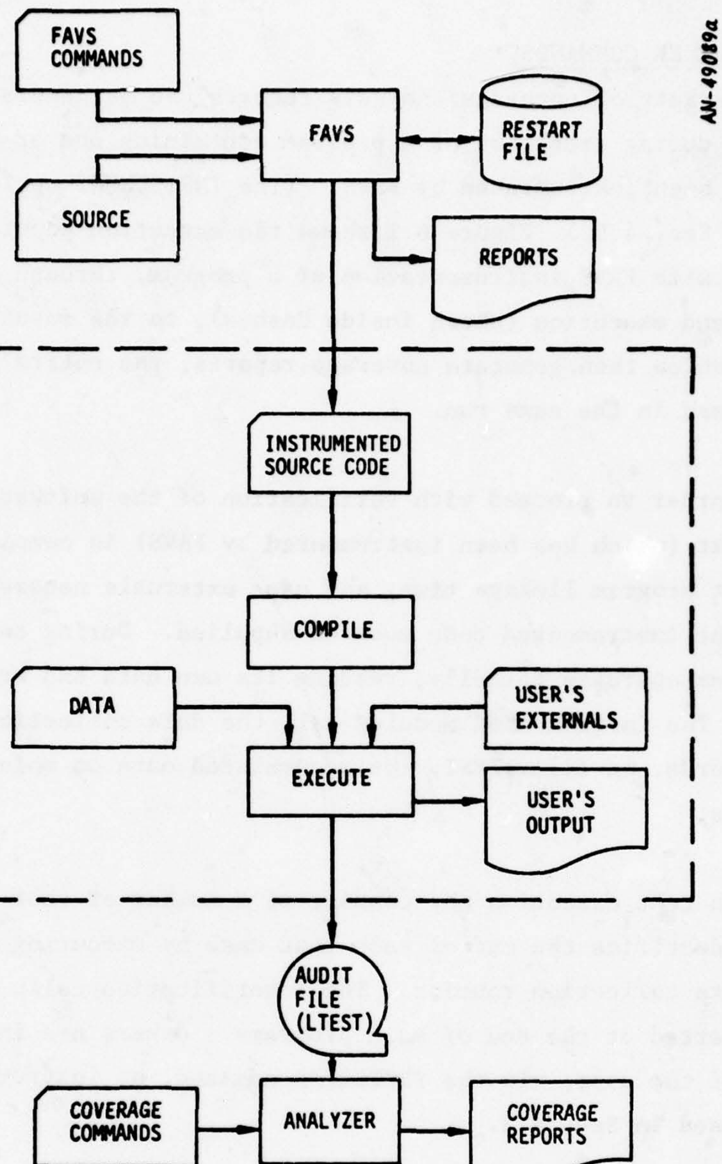
In order to proceed with verification of the software testing, the source text (which has been instrumented by FAVS) is compiled and executed. At program linkage time, any user externals necessary for execution of the instrumented code must be supplied. During test execution the program operates normally, reading its own data and writing its own outputs. The instrumented modules call the data collection routine which records, on file LTEST, the accumulated data on module DD-path traversals.

Each test execution may consist of a number of test cases. The program identifies the end of each test case by executing a special call to the data collection routine. The identification calls are automatically inserted at the end of main programs. Others are inserted by direction of the user, via the TESTBOUND command, at instrumentation time as discussed in Sec. 4.5.

The coverage reports are generated by a set of commands that differ slightly from the FAVS commands (Sec. 3, 4, 5); for this reason the ANALYZER commands are presented in this separate section.

There are two ANALYZER commands, an option selection and a module selection command. The type of report is specified by the command:

```
OPTION(S) = <list>
```




* OPTIONS = INST, STATIC, DOCU, SUMMARY, INPUT, LIST.
OPTION = INST.

† FORTRAN OR DMATRAN SOURCE CODE. DMATRAN INSTRUMENTED SOURCE CODE MUST BE PRECOMPILED BEFORE COMPILATION.
IF DMATRAN IS THE SOURCE CODE LANGUAGE, PRECEDE THE OPTION COMMAND WITH:
LANGUAGE = DMATRAN.

** FOR MODULES = < $NAME_1$ >, < $NAME_2$ >, ..., < $NAME_n$ >.
OPTIONS = SUMMARY, NOTHIT, DETAILED.

**Figure 6.1. Execution Coverage Sequence**

<list> may be one or more of the three options: SUMMARY, NOTHIT, or DETAILED.

If the DETAILED option is specified, then the OPTION command must be preceded by one or more module selection commands:

> FOR MODULE(S) = (<name-1>, <name-2>, ... <name-n>)
> <name> is the name of the module (subroutine, function, or program).

A maximum of 100 modules may be specified at one time. More than one module selection command may be used to accommodate all specified modules. The DETAILED reports will be generated only for the modules named in this command which have been both instrumented and invoked.

Since the Coverage Analysis program records execution trace data in internal tables, the amount of data recorded is limited by table size. The limitations are given below:

| | |
|---|---|
| Maximum number of modules to analyze | 100 |
| Maximum number of test cases | 10 |
| Maximum number of DD-paths to analyze | 2000 |
| Maximum number of DD-paths not traversed in any test case | 1000 |

## 6.1 SUMMARY

The SUMMARY option produces a report which summarizes testing coverage for all instrumented and invoked modules. Figure 6.2 shows a sample SUMMARY report, which lists the following information:

- Test case number
- Module names and numbers of DD-paths
- Number of module invocations, number of DD-paths traversed, and percent coverage for this test case
- Cumulative number of module invocations, number of DD-paths traversed, and percent coverage for all test cases

When multiple test cases are involved, the SUMMARY report shows data from the current test case and the immediately preceding test case. When the end of the trace data is encountered, a cumulative summary of all test cases is produced (Fig. 6.3).

Command

OPTION = SUMMARY

Reports

| | |
|---|---|
| DD-path Summary | (Fig. 6.2) |
| Multiple Test Summary | (Fig. 6.3) |

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

| TEST CASE | MODULE NAME | NUMBER OF D-D PATHS | SUMMARY -- THIS TEST: NUMBER OF INVOCATIONS | D-D PATHS TRAVERSED | PER CENT COVERAGE | CUMULATIVE SUMMARY: NUMBER OF TESTS | INVOCATIONS | TRAVERSED | COVERAGE |
|---|---|---|---|---|---|---|---|---|---|
| 7 | MAIN | 3 | 0 | 1 | 33.33 | 7 | 1 | 2 | 66.67 |
| | CLASS | 98 | 1 | 32 | 32.65 | 7 | 6 | 53 | 54.08 |
| | $$ALL$$ | 101 | | 33 | 32.67 | 7 | | 55 | 54.46 |
| 8 | MAIN | 3 | 0 | 1 | 33.33 | 8 | 1 | 2 | 66.67 |
| | CLASS | 98 | 1 | 37 | 37.76 | 8 | 7 | 53 | 54.08 |
| | $$ALL$$ | 101 | | 38 | 37.62 | 8 | | 55 | 54.46 |

Figure 6.2. DD-Path Summary (with the Immediately Preceding Test Case)

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

| TEST CASE | MODULE NAME | NUMBER OF D-D PATHS | SUMMARY -- THIS TEST: NUMBER OF INVOCATIONS | SUMMARY -- THIS TEST: D-D PATHS TRAVERSED | SUMMARY -- THIS TEST: PER CENT COVERAGE | CUMULATIVE SUMMARY: NUMBER OF TESTS | CUMULATIVE SUMMARY: INVOCATIONS | CUMULATIVE SUMMARY: TRAVERSED | CUMULATIVE SUMMARY: COVERAGE |
|---|---|---|---|---|---|---|---|---|---|
| 1 | MAIN | 3 | 1 | 2 | 66.67 | 1 | 1 | 2 | 66.67 |
| | CLASS | 98 | 0 | 0 | 0.00 | 1 | 0 | 0 | 0.00 |
| | $$ALL$$ | 101 | | 2 | 1.98 | 1 | | 2 | 1.98 |
| 2 | MAIN | 3 | 0 | 1 | 33.33 | 2 | 1 | 2 | 66.67 |
| | CLASS | 98 | 1 | 34 | 34.69 | 2 | 1 | 34 | 34.69 |
| | $$ALL$$ | 101 | | 35 | 34.65 | 2 | | 36 | 35.64 |
| 3 | MAIN | 3 | 0 | 1 | 33.33 | 3 | 1 | 2 | 66.67 |
| | CLASS | 98 | 1 | 2 | 2.04 | 3 | 2 | 35 | 35.71 |
| | $$ALL$$ | 101 | | 3 | 2.97 | 3 | | 37 | 36.63 |
| • | | | • | | | • | | | |
| • | | | • | | | • | | | |
| • | | | • | | | • | | | |
| 9 | MAIN | 3 | 0 | 1 | 33.33 | 9 | 1 | 2 | 66.67 |
| | CLASS | 98 | 1 | 30 | 30.61 | 9 | 8 | 57 | 58.16 |
| | $$ALL$$ | 101 | | 31 | 30.69 | 9 | | 59 | 58.42 |
| 10 | MAIN | 3 | 0 | 1 | 33.33 | 10 | 1 | 2 | 66.67 |
| | CLASS | 98 | 1 | 27 | 27.55 | 10 | 9 | 57 | 58.16 |
| | $$ALL$$ | 101 | | 28 | 27.72 | 10 | | 59 | 58.42 |

Figure 6.3. Multiple Test DD-Path Summary

## 6.2 NOTHIT

The NOTHIT option requests a report which lists DD-paths not executed for all instrumented and invoked modules. Figure 6.4 shows a sample NOTHIT report, which lists the following information:

- Module names
- Test case number
- Number of DD-paths not traversed, for this test case and for all test cases
- DD-path numbers not traversed for this test case and for all test cases

Command

OPTION = NOTHIT

Report

DD-paths Not Executed (Fig. 6.4)

```
==============================================================================================================
 MODULE   I  TEST   I  PATHS  I                              LIST OF DECISION TO DECISION PATHS NOT EXECUTED
 NAME     I NUMBER  I NOT HIT I
==============================================================================================================
<MAIN   > I    10   I     2   I   1    2
          I CUMUL   I     1   I   2
--------------------------------------------------------------------------------------------------------------
<CLASS  > I    10   I    71   I   3    5    6    7   10   12   13   14   15   16   17   18   19   20   21   22   24
                                 32   34   35   36   37   38   39   40   41   42   43   45   48   50   51   52   53
                                 62   65   66   67   68   70   71   73   74   76   77   79   80   81   62   83   84
                                 88   89   90   91   92   93   94   95   96   97   98
          I CUMUL   I    41   I   5    6    7   10   17   28   32   34   36   37   38   40   41   42   43   45   48
                                 53   59   62   67   68   70   71   77   79   80   81   82   83   84   65   86   92
                                 98
--------------------------------------------------------------------------------------------------------------
```

Figure 6.4. DD-Paths Not Executed

## 6.3 DETAILED

The DETAILED option command selects a report which shows a breakdown of individual DD-path coverage. A single testcase report like the one in Fig. 6.5 is generated for each specified module which was instrumented and invoked. Figure 6.6 shows the cumulative report, which is generated after the individual testcase reports. Both provide the following information:

- Module name
- Test case number
- List of DD-path numbers, with an indication of those which were not executed, a graphical representation of the number of executions, and an itemized listing of the number of executions
- Overall module coverage data

**Command**

FOR MODULES = (<name-1>,<name-2>,...<name-n>)
OPTION = DETAILED.

**Reports**

| | |
|---|---|
| Single Test DD-path Execution | (Fig. 6.5) |
| Cumulative DD-path Executions | (Fig. 6.6) |

**Rule**

1. Maximum of 100 modules names specified.
2. Repeat the module selection command as necessary; e.g.,
   FOR MODULES = (<name-1>,...,<name-i>)
   FOR MODULES = (<name-i+1>,...,<name-n>)
3. The module selection command must precede the DETAILED option.

```
RECORD OF DECISION TO DECISION (DD PATH) EXECUTION
----------------------------------------------------------------------------------------------------------
MODULE  $CLASS  $           TEST CASE NO.        8
----------------------------------------------------------------------------------------------------------
DD PATH I  NO.  NOT EXECUTED  I  NUMBER OF EXECUTIONS -- NORMALIZED TO MAXIMUM   I       I  NUMBER OF
NUMBER  I                     I.-------20.-------40.-------60.--------80.-------100.  I       I  EXECUTIONS
----------------------------------------------------------------------------------------------------------
        I                     I
   1    I                     I                                                  I   1   I      1
   2    I                     I                                                  I   2   I      1
   3    I   3       00000     I                                                  I       I
   4    I                     I                                                  I   4   I      1
   5    I   5       00000     I                                                  I       I
  ...   I  ...      00000     I ... ... ...                                      I       I
   7    I   7       00000     I                                                  I       I
   8    I                     I XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX                   I   8   I     48
   9    I                     I XXXXXXXXXXX                                      I   9   I     18
  10    I  10       00000     I                                                  I       I
  11    I                     I XXXX                                             I  11   I      8
  12    I                     I XXXXXX                                           I  12   I     10
  13    I                     I X                                                I  13   I      2
  14    I                     I XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX I  14   I     81
  15    I                     I XXXX                                             I  15   I      8
  16    I                     I XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX     I  16   I     73
  17    I  17       00000     I                                                  I       I
  18    I                     I                                                  I  18   I      1
  19    I                     I XXXX                                             I  19   I      7
  20    I  20       00000     I                                                  I       I
                              .                                                  .
                              .                                                  .
                              .                                                  .
  69    I                     I                                                  I  69   I      1
  70    I  70       00000     I                                                  I       I
  71    I  71       00000     I                                                  I       I
  72    I                     I                                                  I  72   I      1
  73    I                     I                                                  I  73   I      1
  74    I  74       00000     I                                                  I       I
  ...   I  ...      00000     I ... ... ...                                      I       I
  98    I  98       00000     I                                                  I       I
----------------------------------------------------------------------------------------------------------
                                                  TOTAL NUMBER OF DD PATH EXECUTIONS =     422

  TOTAL OF 61  NOT EXECUTED      EXECUTED  37/ 98        PERCENT EXECUTED =     37.76
```

Figure 6.5. Single Test DD-Path Execution

```
RECORD OF DECISION TO DECISION (DD PATH) EXECUTION

------------------------------------------------------------------------------------------------------------
MODULE  $CLASS  $          CUMULATIVE RESULTS OF    10 TEST CASES

------------------------------------------------------------------------------------------------------------
DD PATH I  NO.  NOT EXECUTED  I   NUMBER OF EXECUTIONS -- NORMALIZED TO MAXIMUM         I         I  NUMBER OF
NUMBER  I                     I.-------20.-------40.-------60.-------80.-------100.     I         I EXECUTIONS
------------------------------------------------------------------------------------------------------------
        I                     I
   1    I                     I X                                                       I    1    I        9
   2    I                     I                                                         I    2    I        8
   3    I                     I                                                         I    3    I        1
   4    I                     I                                                         I    4    I        8
   5    I   5          00000  I                                                         I         I
  ...   I  ...         00000  I ... ... ...                                             I         I
   7    I   7          00000  I                                                         I         I
   8    I                     I XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX                     I    8    I      304
   9    I                     I XXXXXXXXXXXXXX                                          I    9    I      118
  10    I  10          00000  I                                                         I         I
  11    I                     I XXXXXXXXXXX                                             I   11    I       95
  12    I                     I XX                                                      I   12    I       23
                              .                                                         .
                              .                                                         .
                              .                                                         .
        I                     I
  88    I                     I                                                         I   88    I        1
  89    I                     I                                                         I   89    I        1
  90    I                     I XXXXX                                                   I   90    I       49
  91    I                     I X                                                       I   91    I       12
  92    I  92          00000  I                                                         I         I
  93    I                     I X                                                       I   93    I       13
  94    I                     I X                                                       I   94    I       13
  95    I  95          00000  I                                                         I         I
  ...   I  ...         00000  I ... ... ...                                             I         I
  98    I  98          00000  I                                                         I         I
------------------------------------------------------------------------------------------------------------

                                                   TOTAL NUMBER OF DD PATH EXECUTIONS =     2511

   TOTAL OF 41  NOT EXECUTED     EXECUTED  57/ 98       PERCENT EXECUTED =     58.16
```

**THIS PAGE IS BEST QUALITY PRACTICABLE FROM COPY FURNISHED TO DDC**

Figure 6.6. Cumulative DD-path Execution

# APPENDIX A

# FAVS SEGMENT COMMANDS

FAVS processing is controlled by different types of segment commands which are listed below. Appendix B presents a complete description, in alphabetical order, of the FAVS segment commands.

- Library Commands

  NEW LIBRARY = <name>.

  OLD LIBRARY = <name>.

  R/W LIBRARY = <name>.

- Start-up Commands

  FILENAME,LOG=<file-number>.

  FILENAME,PUNCH=<file-number>.

  LANGUAGE = FORTRAN/DMATRAN.

  SEGMENTS.

  START.

- Process Option Commands

  INSTRUMENT,TESTBOUND,MODULE = ( name ), STATEMENT = <number>.

  STRUCTURAL, COMPUTE = FULL.

  STRUCTURAL, JUNCTION = ON.

- Module Selection Commands

  MODULE = ( name ).

  FOR MODULE = (<name-1>,...,<name-n>).
  END FOR.

  FOR ALL MODULES.
  END FOR.

- <u>Process Execution Commands</u>

  ASSIST,REACHING SET,TO = <DD-path number>,
  FROM = <DD-path number>,{ITERATIVE}.

  BASIC.

  BUILD,DMT .

  BUILD,PREDICATE.

  BUILD,PARMETERS.

  BUILD,CROSS.

  DOCUMENT,BANDS.

  DOCUMENT,BANDS = <number>.

  DOCUMENT,COMMONS,PRINT = FULL.

  DOCUMENT,COMMONS,PRINT = PART.

  DOCUMENT,COMMONS,PRINT = SUMMARY.

  DOCUMENT,CROSSREF.

  DOCUMENT,INVOKES.

  DOCUMENT,MATRIX,LIBRARY.

  DOCUMENT,READ.

  INSTRUMENT.

  STRUCTURAL.

- <u>Standard Print Commands</u>

  PRINT,DDPATHS.

  PRINT,MODULE.

  PRINT,PROFILE.

- <u>Run Termination Command</u>

  END .

## A.1 LIBRARY COMMANDS

The LIBRARY commands define the name and status of library to be used by FAVS.

- NEW LIBRARY = <name> is implicitly generated by default whenever an OLD LIBRARY command is not supplied. It causes a new library to be created during BASIC and STRUCTURAL processing.
- OLD LIBRARY = <name> informs FAVS that an old library is being supplied, so it is not necessary to execute the BASIC and STRUCTURAL steps for the current run.
- R/W LIBRARY = <name> informs FAVS that an old library is being supplied and that additional source will be added to it.

If a LIBRARY command is used, it must be supplied before the START command.

## A.2 STARTUP COMMANDS

The FILENAME,LOG = <file number> command is used to direct the activity log produced during FAVS processing to an appropriate file.

The LANGUAGE = <name> command identifies the language of the source code to be analyzed. The possibilities for <name> are:

- FORTRAN (default)
- DMATRAN

This command must precede the START command.

The SEGMENT command causes the segment commands generated from user commands to be written on LOUT.

The startup command terminates the library description and indicates the start of processing. The command is:

START.

A.3 PROCESS OPTION COMMANDS

Processing steps STRUCTURAL and INSTRUMENT have option commands which define the action taken when the process execution command is given. The process option commands follow the START command and are followed by the appropriate process execution commands (see Sec. A.5).

A.3.1 STRUCTURAL Option Commands

The following commands generate additional structural information during the STRUCTURAL analysis. They are never used with any option other than the RESTRUCTURE option.

STRUCTURAL, COMPUTE = FULL.

STRUCTURAL, JUNCTION = ON.

A.3.2 INSTRUMENT Option Commands

The following command is used to identify test case boundaries in instrumented source code:

```
INSTRUMENT,TESTBOUND,MODULE = (<name>),
          STATEMENT = <number>.
```

The <number> is the FAVS number of the statement at which one testcase is to end and a second is to begin.

A.4 MODULE SELECTION COMMANDS

Many FAVS commands require specification of the particular modules upon which computations are to be performed. Some of the standard print commands require this, for example. Commands are available to select a single module, a subset of modules, or all modules in a library. If two or more versions of a module appear on a library, the last one entered on the library will be selected.

A.4.1 Single Module Selection

The following command selects a single module:

```
MODULE = (<name>).
```

All subsequent commands (if they refer to a specific module) are applied to this single module. There can be any number of MODULE = commands.

### A.4.2 Selected Module Iteration

The following sequence selects a subset of modules, by name, and iterates a block of commands (which cannot contain another iteration) once for each specified module:

```
FOR MODULE = (<name>,<name-2>,...<name-n>).
(commands)
END FOR.
```

### A.4.3 All-Modules Iteration

The following sequence selects each known module within the current library and iterates a block of commands (which cannot contain another iteration) once for each known module.

```
FOR ALL MODULES.
(commands)
END FOR.
```

## A.5 PROCESS EXECUTION COMMANDS

The process execution commands for the FAVS processing steps are:

```
ASSIST,REACHING SET,TO = <DD-path number>,
     FROM = <DD-path number>,{ITERATIVE}.
BASIC.
BUILD,DMT .
BUILD,PREDICATE.
BUILD,PARMETERS.
BUILD,CROSS.
DOCUMENT,BANDS.
DOCUMENT,BANDS = <number>.
DOCUMENT,COMMONS,PRINT = FULL.
DOCUMENT,COMMONS,PRINT = PART.
DOCUMENT,COMMONS,PRINT = SUMMARY.
DOCUMENT,CROSSREF.
```

DOCUMENT,INVOKES.

DOCUMENT,MATRIX,LIBRARY.

DOCUMENT,READ.

INSTRUMENT.

STRUCTURAL.

Each of these commands (except for BASIC and BUILD, PARMETERS and BUILD, CROSS and DOCUMENT,CROSSREF) causes execution of the processing step on a previously selected set of modules. The exceptions cause the processing to apply to all the modules in the library.

## A.6 STANDARD PRINT COMMANDS

The standard print commands provide the means to generate formatted output of FAVS internal tables. These print commands are universal; i.e., they can be used in any processing step. The standard print commands are of the form:

PRINT,<table-name>.

where <table-name> is DDPATHS, MODULE, or PROFILE. For a PRINT command to be accepted, a set of modules must have been selected with a module selection command (see Sec. A.4).

## A.7 RUN TERMINATION COMMAND

A FAVS run terminates on the END command, which provides for correctly closing any files.

The run termination command is:

END.

## A.8 ORDER AND USE OF THE SEGMENT COMMANDS

The segment commands should be used in the order in which they have been presented in this section. The LIBRARY commands (if present) are first, followed by the start-up commands, etc. The run termination command is always the last command.

Each of the twelve different FAVS reports that can be generated by the OPTION command is listed along the top in Table A.1. On the left side are listed the segment FAVS commands in the order they are needed to generate the report named at the top of the column. For example, if the user wants only two of the five reports produced by the DOCUMENT option, this table shows which segment commands must be used and the order in which they should be placed.

The module selection commands must be supplied by the user. (See Sec. A.4.) A module selection command is required only for following commands:

BUIL,DMT.
BUIL,PRED.
STRU.
PRIN,MODU.
PRIN,DDPA.
STAT.
DOCU,INVO.
DOCU,BAND.
PRIN,PROF.

TABLE A.1

SEGMENT COMMANDS REQUIRED TO GENERATE FAVS REPORTS

Name of Report:

| SEGMENT COMMANDS: | STATEMENT LISTING | COMMONS MATRIX | LIBRARY DEPENDENCE MATRIX | STATEMENT PROFILE | COMMONS MATRIX (ENHANCED) | CROSS REFERENCE | INVOCATION BANDS | INVOCATION SPACE | READ STATEMENTS | STATIC ANALYSIS | DD-PATH DEFINITIONS | REACHING SET ANALYSIS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STAR. | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| BASI.* | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| BUIL,DMT.* | | ● | ● | | ● | ● | ● | ● | | ● | | |
| BUIL,PRED.* | | | | | | | | | | ● | | |
| BUIL,PARM. | | ● | | | ● | ● | | | | ● | | |
| BUIL,CROS. | | ● | | | ● | ● | | | | ● | | |
| STRU. | | | | | | | | | | ● | ● | ● |
| PRIN,MODU. | ● | | | | | | | | | | | |
| PRIN,DDPA. | | | | | | | | | | | ● | |
| STAT. | | | | | | | | | | ● | | |
| DOCU,INVO. | | | | | | | | ● | | | | |
| DOCU,BAND. | | | | | | | ● | | | | | |
| PRIN,PROF. | | | | ● | | | | | | | | |
| DOCU,MATR,LIBR. | | | ● | | | | | | | | | |
| DOCU,COMM,PRIN=PART. | | ● | | | | | | | | | | |
| DOCU,COMM,PRIN=SUMM. | | | | | ● | | | | | | | |
| DOCU,READ. | | | | | | | | | ● | | | |
| DOCU,CROS. | | | | | | ● | | | | | | |
| ASSI,REAC,TO=.... | | | | | | | | | | | | ● |
| END . | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | LIST | ← SUMMARY → | | | ← DOCUMENT → | | | | | STATIC | INSTRUMENT | REACHING SET |

*Do not use if RESTART or OLD LIBRARY has been selected.

## APPENDIX B

## DETAILED DESCRIPTION OF FAVS SEGMENT COMMANDS

 ASSIST,REACHING SET,<specs>.

Description

This command is used to analyze the flow required to reach a particular DD-path according to the items in hte specification list. The reaching set consists of all DD-paths which flow between the beginning and ending DD-paths. There are three types of specifications which may be present in any order but must be separated by commas:

TO = <DD-path number>
FROM = <DD-path number>
ITERATIVE

The path for the reaching set (i.e., the target of flow) is named by the (required) specification TO = <DD-path number>. In the absence of the FROM = <DD-path number> specification, the flow is assumed to start with the first executable statement in the module. The FROM specification allows the user to identify the DD-path where flow starts. The analysis begins with the first executable statement on the DD-path.

The (optional) ITERATIVE specification allows the user to control the set of DD-paths in the analysis. If ITERATIVE is not specified, all flows which include iteration are suppressed in determination of paths of control. If ITERATIVE is specified, the flows include iteration.

Rules

1. Maximum of 100 DD-paths per reaching set path.
2. Maximum of 100 outways per decision.
3. Maximum of 1600 DD-paths per analyzed module for reaching set.
4. Maximum of 3200 statements per analyzed module for reaching set.
5. Maximum of 200 statements in reaching set.

 ASSIST,REACHING SET,<specs>.

## Sample Output

```
REACHING SET ANALYSIS                    SUBROUTINE CLASS ( K, ITYP )

              NON-ITERATIVE REACHING SET FROM DD-PATH   8 TO DD-PATH 40
           DDPATHS IN REACHING SET
                         8     9    10    11    12    13    14    15    17    18    22
              23    24    25    26    27    28    36    39    40    43    44    47
           SOURCE CODE IN REACHING SET
--------------------------------------------------------------------------------
                             . . .
      73 ( 1)                .  JCH = K ( J )
      74 ( 1)                .  IF ( JCH .EQ. KBLNK )                 (  8-  9)
      75 ( 2)                .  .  GOTO 26
      76 ( 1)                .  IF ( JHOLL ) 12, 12, 7                ( 10- 12)
      77 ( 1)         7      .  DO 8 L = 1, 10
      78 ( 2)                .  .  IF ( JCH .EQ. KDEC ( L ) )         ( 13- 14)
      79 ( 3)                .  .  .  GOTO 10
      80 ( 1)         8      .  CONTINUE                              ( 15- 16)
      81 ( 1)                .  IF ( JHOLL - 1 ) 11, 11, 9            ( 17- 19)
                             . . .
      84 ( 1)        10      .  JHOLL = JHOLL + 1
      85 ( 1)                .  GOTO 25
      86 ( 1)        11      .  JHOLL = 0
      87 ( 1)        12      .  IF ( JCH .EQ. KLPAR )                 ( 22- 23)
      88 ( 2)                .  .  GOTO 20
      89 ( 1)                .  IF ( JCH .EQ. KRPAR )                 ( 24- 25)
      90 ( 2)                .  .  GOTO 18
      91 ( 1)                .  IF ( JCH .EQ. KCMA )                  ( 26- 27)
      92 ( 2)                .  .  GOTO 22
      93 ( 1)                .  IF ( JCH .EQ. KEQ )                   ( 28- 29)
      94 ( 2)                .  .  GOTO 23
                             . . .
     100 ( 1)        18      .  JSW = JSW - 1
     101 ( 1)                .  IF ( JSW ) 19, 19, 25                 ( 34- 36)
                             . . .
     104 ( 1)        20      .  JSW = JSW + 1
     105 ( 1)        21      .  JHOLL = 1
     106 ( 1)                .  GOTO 25
     107 ( 1)        22      .  IF ( JSW ) 30, 30, 21                 ( 37- 39)
     108 ( 1)        23      .  IF ( JSW ) 24, 24, 32                 ( 40- 42)
     109 ( 1)        24      .  JEQ = 1
     110 ( 1)        25      .  IF ( ISW ) 26, 26, 27                 ( 43- 45)
     111             26      CONTINUE                                 ( 46- 47)
                             . . .
--------------------------------------------------------------------------------
```

## Description

The BASIC command causes BASIC syntax analysis to be executed. The action performed by the BASIC command is the construction of a new library containing a module descriptor block, a statement descriptor table, a statement table, a symbol locator table, and a symbol table for each module on the INPUT file. The output is written on the file LOG and may be obtained by using the command FILENAME, LOG = <file name> to equivalence files.

## Rules

1. See syntax constraints in Sec. 4.2.
2. Maximum of 250 modules can be on the library.

## Sample Output

```
FIRST PASS OF NEW MODULE BEGUN
CURRENT CPU TIME =        .262
SOURCE TEXT MODIFIED FOR SYSTEM INPUT
FIRST PASS OF NEW MODULE COMPLETED
CURRENT CPU TIME =        .266          TIME SINCE LAST CHECK =      .003

SECOND PASS OF NEW MODULE BEGUN
CURRENT CPU TIME =        .266
<MAIN    > IS MODULE BEING PROCESSED
STATEMENT BLOCKS AND STATEMENT DESCRIPTOR BLOCKS GENERATED
SECOND PASS OF NEW MODULE COMPLETED
CURRENT CPU TIME =        .284          TIME SINCE LAST CHECK =      .018

THIRD PASS OF NEW MODULE BEGUN
CURRENT CPU TIME =        .284
SYMBOL TABLE BLOCKS GENERATED
INTER-STATEMENT POINTER GENERATED
THIRD PASS OF NEW MODULE COMPLETED
CURRENT CPU TIME =        .290          TIME SINCE LAST CHECK =      .007
```

BUILD,DMT .

## Description

This command builds a Dependent Module Table for each module specified by a module selection command. The table contains the names of modules invoked, the statement number of the invocation, the number of times a module is invoked, the length of the invokee's name, and whether invokee is an internal procedure (i.e., a DMATRAN BLOCK).

## Rule

This command must be used with a module selection command; otherwise, the table is built only for the last module on the library.

This command must precede any of the processing execution commands with one exception; it should follow the BASIC command.

Description

This comamnd builds tables containing information about the predicates in a module; i.e., the length and origin of each predicate.

Rule

This command must be used with a module selection command; otherwise, the table is built only for the last module on the library.

This command must precede any of the processing execution commands with one exception; it should follow the BASIC command.

Description

This command generates information about parameters (number, location, etc.) and stores it in tables in the database library. All parameters encountered in any of the modules are included in the table.

Rule

This command must precede any of the processing execution commands with one exception; it should follow the BASIC command.

No module selection command is required.

## Description

This command builds the Symbol Cross Reference Table which lists symbols encountered in any of the modules on the library. The following information is stored with each symbol:

- Original symbol entry
- Names of modules containing the symbol
- Statement numbers where symbol occurs
- Use of symbol

## Rule

This command must be the last BUILD command if any others are used. It follows the BASIC command, but precedes all the other processing execution commands.

No module selection command is necessary.

## Description

This command has two forms:

DOCUMENT,BANDS. or

DOCUMENT,BANDS = <number>. (DEFAULT <number> = 5)

The outcome of this command is a "snapshot" of the position of the selected module within the intermodule hierarchy. The sample output shows an example of the BANDS report. To the left of the selected module is shown the structure of the calls to the module; to the right of the selected module is shown the invocation structure emanating from the module. The number of bands is the width (in each direction) of the structure displayed. Up to five bands may be displayed on this report. This report is useful in determining the extent of intermodule dependence to several levels. Modules which are called from only one other module are potential candidates to head a segment for overlay purposes.

The modules listed under column -1 call the selected module directly (STRUCT in the sample output below), while the modules shown under column -2 call those in column -1, etc. Modules listed under column 1 are called by the selected module (CONT, KEMPTY, and PUTFTN in the sample output), while those under column 2 are called by those listed under column 1, etc.

## Rule

Maximum bandwidth is 5.

## Sample Output

```
INVOCATION BANDS              SUBROUTINE FULCON ( LABEL )
TO LEVEL 2
LEVEL      -5      -4      -3      -2      -1       0       1       2       3       4       5
-----------------------------------------------------------------------------------------------
                                                  FULCON
                                          STRUCT          CONT
                                  CONTRL                          MOVE=D
                                                          KEMPTY
                                                                  KCOMP
                                                          PUTFTN
                                                                  INCENT
                                                                  MOVE=D
                                                                  MOVE=U
                                                                  SPRY=D
-----------------------------------------------------------------------------------------------
```

 DOCUMENT,COMMONS,PRINT=FULL.

Description

This command generates two matrixes. The Library Common Block Matrix lists all the common blocks encountered in any of the modules in the set that was analyzed. An "X" indicates that at least one of the variables in the common blocks was used. An "O" indicates that no symbol was ever referenced in the module.

The Library Common Symbol Matrix lists all the symbols in each of the common blocks. The number of the common block (as assigned in the first matrix) is printed to the far left of the name of the symbol.

Sample Output

The output from this command includes all COMMON symbols; thus the report generated by the command DOCUMENT,COMMONS,PRINT=SUMMARY (page B-12) is a subset of the PRINT=FULL option.

The FULL option report is used to identify COMMON variables which can be removed from COMMON blocks. The entries in the COMMON symbol matrix for unreferenced variables are "O's" for all modules containing the COMMON block.

## DOCUMENT,COMMONS,PRINT=PART

DOCUMENT,COMMONS,PRINT=PART.

### Description

This report lists all modules and all common blocks encountered. An "X" indicates the presence of that common in a module.

### Sample Output

```
CCMMONS MATRIX

LIBRARY COMMON BLOCK MATRIX

------------------------.---------
C **         *          .        *
O * * MODULE * C C E F K.M M P S *
M *  *       * O O X U E.A O U T *
M *    *     * N N A L M.I V T R *
O *      *   * T T M C P.N E F U *
N *       *  * R   P O T.  W T C *
  *        * * L   L N Y.  D N T *
N *        * *          .        *
O * COMMON * *          .        *
  *         **          .        *
------------------------.---------
1  * ACCTNG  * X        .        *
2  * CARDS   * X       X.        *
3  * CCNSTN  * X X     X.    X X *
4  * FORTRN  * X X      .    X X *
5  * INTERN  * X     X  .    X X *
6  * INVOKE  * X        .      X *
7  * RECNIZ  * X        .        *
8  * SESE    * X        .        *
9  * STACK   * X        .      X *
10 * STATE   * X     X  .    X X *
11 * STYPE   * X        .      X *
12 * TRACE   * X        .    X   *
13 * USEOPT  * X X     X.    X X *
14 * WARNIN  * X        .        *
------------------------.---------
```

## Description

Two matrices are produced by this report. The first one lists all common blocks encountered in any one of the modules in the set which was analyzed. *If at least one symbol was used,* it is indicated with an "X". If no symbol was ever references in the module, this is indicated by an "O". Routines from which a common block may safely be removed are easily found.

The second matrix lists only the symbols which are used by some module; the number of the common block in which it is found is printed to the left and corresponds to the number given to the common block in the first matrix. This report is an excellent aid when changes are being made to a software system.

## Sample Output

LIBRARY COMMON BLOCK MATRIX

| COMMON NO. | MODULE / COMMON | CONTRL | CONT | EXAMPL | FULCON | KEMPTY. | MAIN. | [illegible] | PLTFIN | STRUCT |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ACCTNG | O | | | | | | | | |
| 2 | CARDS | X | | | | O | | | | |
| 3 | CONSTN | O | X | | | X | | | X | X |
| 4 | FORTRN | O | X | | | | | | X | O |
| 5 | INTERN | X | | | O | | | | X | O |
| 6 | INVOKE | O | | | | | | | | X |
| 7 | RECNIZ | O | | | | | | | | |
| 8 | SESE | O | | | | | | | | |
| 9 | STACK | O | | | | | | | | X |
| 10 | STATE | X | | | X | | | | O | X |
| 11 | STYPE | O | | | | | | | | O |
| 12 | TRACE | X | | | | | | | X | |
| 13 | USEOPT | X | O | | | O | | | X | X |
| 14 | WARNIN | O | | | | | | | | |

LEGEND

COMMONS VS. MODULES

X => AT LEAST ONE SYMBOL REFERENCED
O => NO SYMBOL EVER REFERENCED

SYMBOLS VS. MODULES

X => SYMBOL SET AND USED
O => SYMBOL NEVER SET OR USED
S => SYMBOL SET ONLY
U => SYMBOL USED ONLY
E => SYMBOL EQUIVALENCED (OVERLAID) ONLY
A => SYMBOL IS AN ARRAY

LIBRARY COMMON SYMBOL MATRIX

| COMMON NO. | MODULE / SYMBOL | CONTRL | CONT | EXAMPL | FULCON | KEMPTY. | MAIN. | [illegible] | PLTFIN | STRUCT |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | IECF | U | | | | O | | | | |
| 13 | INCON | O | O | | | O | | | U | O |
| 9 | INSTAK | C | | | | | | | | X |
| 10 | ITYPE | U | | | O | | | | O | U |
| A4 | KABEL | C | X | | | | | | U | O |
| 4 | KENGTH | O | S | | | | | | X | O |
| A9 | KFIN | O | U | | | | | | U | O |
| 13 | KOMFTN | O | O | | | O | | | U | U |
| 5 | KSTMT | U | | | O | | | | U | O |
| A10 | LABEL | O | | | X | | | | O | U |
| 3 | LEX | C | U | | | U | | | U | U |
| 10 | LENGTH | S | | | O | | | | C | O |
| 10 | LINBEG | U | | | O | | | | O | O |
| 10 | LINENC | L | | | O | | | | O | O |
| A10 | LIST | O | | | O | | | | O | U |
| 10 | LPOINT | C | | | U | | | | O | U |
| A5 | LSTACK | O | | | | | | | | X |
| 10 | LTYPE | X | | | O | | | | J | U |
| 13 | LUNFOR | O | O | | | O | | | U | O |
| 13 | LUNOUT | U | O | | | O | | | O | O |
| 10 | MENGTH | C | | | O | | | | O | U |
| 12 | NALTER | C | | | | | | | U | |
| A6 | NAMES | O | | | | | | | | U |
| 5 | NFATER | U | | | O | | | | O | O |
| 13 | NINONT | C | O | | | O | | | X | O |
| 5 | NLINES | S | | | O | | | | O | O |
| 6 | NODE | O | | | | | | | | X |
| A6 | NCPLOK | O | | | | | | | | S |
| A6 | NOINV | O | | | | | | | | X |
| 10 | NSTATE | X | | | O | | | | O | O |

## Description

The outcome of this command is a cross-reference listing of names in the entire library and their usage. The names listed are all items, files, switch names, labels, and subprogram names.

## Rule

A maximum of 100 modules can be used in cross-reference mapping.

## Sample Output

```
CROSS REFERENCE

GENERAL CROSS REFERENCE LISTING

MODULES INCLUDED --
          CONTRL
          CONT
          EXAMPL
          FULCON
          KEMPTY
          MAIN
          MOVEWD
          PUTFTN
          STRUCT

SYMBOL    MODULE    USED/SET/DEFINITION ( * INDICATES SET, D INDICATES DEFINITION )
ACT1      CONTRL     172
ACT2      CONTRL     174
ASSIGN    STRUCT     180
BGSCAN    CONTRL     168
CONTRL    CONTRL       1
          MAIN         2
CONT      CONT         1
          FULCON      14
          STRUCT      86  103  124  153  165  202  236  258  262  292  303  306  345
ENDER     CONTRL     183
ERROR     STRUCT      53  107  111  113  128  130  169  171  213  217  219  240  244
EXAMPL    EXAMPL       1
          MOVEWD      33
FULCON    FULCON       1
          STRUCT      84  101  122  137  160  199  234  255  275  298
GENASS    STRUCT     341
GENGO     STRUCT     369
GENLAB    STRUCT      73   81   85   98  102  123  139  141  149  152  161  164  195
                     281  283  291  299  302  305  339  340  357  360  371
GENVAR    STRUCT     179  208
GETSTM    CONTRL     164
GOTO      STRUCT      82   99  150  162  196  232  278  300  343  358
IARRY1    MOVEWD       1   23D  29*
IARRY     MOVEWD       1   22D  29
ICONT     CONT       24D  25D  25D  25D  25D  25D  25D  25D  25D  28
IEOF      CONTRL     29D 165  180
          KEMPTY      5D
IERROR    STRUCT      92*  93   94*  95  110  120* 121  127  158* 159  168  190* 191
                     243  253* 254  265  296* 297  309
```

## Description

The outcome of this module command is a report which shows (1) the invocations of the selected module from all other known modules, and (2) the invocations within the selected module to all other modules. The sample output shows a report produced by this command. For each module the FAVS statement number of the invocation and the source text for the invocation are shown.

## Sample Output

```
INVOCATION SPACE                              SUBROUTINE CONT ( LABEL )
-----------------------------------------

INVOCATIONS FROM WITHIN THIS MODULE
-----------------------------------------

MODULE MOVEWD
STMT =   26           CALL MOVEWD ( 5 , 1 , LABEL , 1 , KABEL )
STMT =   28           CALL MOVEWD ( 8 , 1 , ICONT , 1 , KFTN )

INVOCATIONS TO THIS MODULE FROM WITHIN LIBRARY
-----------------------------------------

MODULE FULCON
STMT =   14           CALL CONT ( LABEL )

MODULE STRUCT
STMT =   86           CALL CONT ( LAB )
STMT =  103           CALL CONT ( LAB )
STMT =  124           CALL CONT ( LAB )
STMT =  153           CALL CONT ( LAB )
STMT =  165           CALL CONT ( LAB )
STMT =  202           CALL CONT (LAB )
STMT =  236           CALL CONT ( LAB )
STMT =  258           CALL CONT (LAB )
STMT =  262           CALL CONT ( LAB )
STMT =  292           CALL CONT ( LAB )
STMT =  303           CALL CONT ( LAB )
STMT =  306           CALL CONT (LAB )
STMT =  345           CALL CONT ( NAME1 )
STMT =  361           CALL CONT ( NAME1 )
STMT =  373           CALL CONT ( NAME1 )
```

## Description

This module report shows all invocations, along with the statement numbers, to and from the specified module. It is useful in examining actual parameter usage.

## Sample Output

```
LIBRARY DEPENDENCE MATRIX

****************************************************
** INVOKEE *         *                             *
* *        *CCEFKMMPS*AAABEEGGGGGGIIIIIIIKKMNNNNPSV*
*  *       *OOXUEAOUT*CCSGNREEEEEOFFGNNNWCLOADEEUPE*
*   *      *NNALMIVTR*TTSSDRNNNNTTCSRDDIIOAVMSWWTRR*
*    *     *TTMCPNEFU*12ICEOAGLVSOAOOELTTMSEOCLPIYB*
*     *    *R POT WTC*  GARRSCAAT S UNEAHPSWBAAAFWA*
*      *   *L LNY ONT*  NN  S BRM E PTVLN 1U NBG DT*
*       *  *         *                             *
*        * *         *                             *
* INVOKER **         *                             *
****************************************************
* CONTRL   **       X*XX XX     X      X     X XX  *
* CONT     * *    X  *                             *
* EXAMPL   *  *      *                             *
* FULCON   * X *X  X *                             *
* KEMPTY   *    *    *                   X         *
* MAIN     *X    *   *                             *
* MOVEWD   *  X   *  *                             *
* PUTFTN   *      X* *               X     X     X *
* STRUCT   * X X   X**  X  XXXXX XXXX X X X X X  XX*
****************************************************

THE FOLLOWING MODULES ARE NOT INVOKED BY ANY MODULE ON THE LIBRARY

      MAIN

THE FOLLOWING MODULES DO NOT INVOKE ANY MODULE ON THE LIBRARY

      EXAMPL      KEMPTY
```

## Description

This report provides a list of all the program modules in which a READ appears. The source statements are reproduced along with the defining FORMAT. This report may be used to locate all the points where variables are being input to the system.

## Sample Output

```
READ STATEMENTS


THE FOLLOWING MODULES CONTAIN READ STATEMENTS

              GETCRD
              GETINS
-----------------------------------------------------------------

READ STATEMENTS AND ASSOCIATED FORMATS
--------------------------------------

--- GETCRD  ---
  16                     READ ( LUNIN, 1 ) ( LCARD ( I ), I = 1, 80 )
  17           1         FORMAT (80A1)

--- GETINS  ---
  44                     READ ( 5, 1 ) ( NUN ( I ), I = 1, NOPTS )
  45           1         FORMAT (12I5 )
-----------------------------------------------------------------
```

Description

After the operations indicated by the contents of the command file are complete, FAVS must be "shut down" by use of the following command:

END.

The END command must always be the last FAVS command. The actions which occur as a consequence of this command are important to the FAVS user in only one regard--when the library is completed and is to be saved for future runs. The wrapup sequence provides necessary information sbout the contents of the new library. The sample output shows the standard wrapup output. This report contains the Module Descriptor Table, library information, and Interface Use Statistics.

The wrapup output is written on the file LOG. By using the FILENAME command to equivalence the LOG and OUTPUT files, the user may obtain a printout of this information.

Rule

This must be the last FAVS command.

END. END.

## Sample Output

SYSTEM WRAPUP...

MODULE DESCRIPTOR BLOCKS AS THEY ARE CURRENTLY KNOWN..

| NO. | NAME | TYPE | MODE | LANGUAGE DIALECT | PARENT | STMTS | EXEC STMTS | 1ST EXEC | ARGS | ENTRS | SYMS | DCPS | INVOKES | LONG NAME |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | BSORT | SUBROUTINE | TYPELESS | FORTRAN | | 20 | 19 | 3 | 2 | 1 | 12 | 9 | 0 | |

LIBRARY HEADER ---

| | |
|---|---|
| MODE OF ACCESS | NEW |
| FRAGMENT SIZE | 500 |
| LIBRARY SIZE | 7000 |
| NUMBER OF MODULES | 1 |
| NUMBER OF ENTRIES | 1 |
| NUMBER OF FRAGMENTS | 13 |
| NUMBER OF TOKENS. | 260 |

INTERFACE USE STATISTICS....

| NO. | NAME | TYPE | SCOPE | GETWRDS | GETBLKS | PUTWRDS | PUTBLKS | SEARCHES | ACTIVATES | SWITCHES | FETCHES |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | MLT | PERM | SYS | 67 | 0 | 6 | 0 | 0 | 0 | 0 | 0 |
| 2 | FCT | PERM | SYS | 147 | 0 | 41 | 0 | 0 | 0 | 0 | 0 |
| 3 | EPT | PERM | SYS | 0 | 3 | 0 | 1 | 0 | 1 | 2 | 2 |
| 4 | TKI | PERM | SYS | 192 | 0 | 260 | 0 | 223 | 1 | 2 | 2 |
| 5 | TOK | PERM | SYS | 467 | 0 | 320 | 260 | 0 | 1 | 72 | 2 |
| 6 | TKL | PERM | SYS | 0 | 0 | 0 | 260 | 0 | 1 | 2 | 2 |
| 7 | WAIT | PERM | SYS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | WMAX | PERM | SYS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | AMAX | PERM | SYS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | AXIO | PERM | SYS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | MCNG | TEMP | SYS | 0 | 2 | 0 | 2 | 4 | 1 | 1 | 1 |
| 36 | MCB | PERM | MOD | 4 | 0 | 0 | 4 | 0 | 1 | 1 | 1 |
| 37 | SDB | PERM | MOD | 335 | 120 | 83 | 45 | 3 | 1 | 1 | 1 |
| 38 | SB | PERM | MOD | 0 | 26 | 0 | 40 | 0 | 1 | 1 | 1 |
| 39 | STB | PERM | MOD | 0 | 46 | 0 | 50 | 1 | 1 | 1 | 1 |
| 40 | TB | PERM | MOD | 0 | 49 | 0 | 20 | 0 | 1 | 1 | 1 |
| 41 | CPT | PERM | MOD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 42 | PRED | PERM | MOD | 0 | 0 | 0 | 5 | 0 | 1 | 1 | 1 |
| 43 | PDAX | PERM | MOD | 5 | 0 | 15 | 0 | 0 | 1 | 1 | 1 |
| 44 | CCP | TEMP | MOD | 39 | 9 | 18 | 9 | 29 | 1 | 1 | 1 |
| 45 | CS | TEMP | MOD | 6 | 4 | 0 | 9 | 0 | 1 | 1 | 1 |
| 46 | LBL | TEMP | MOD | 5 | 0 | 0 | 0 | 5 | 1 | 1 | 1 |
| 47 | ACM1 | TEMP | MOD | 45 | 0 | 27 | 9 | 0 | 1 | 1 | 1 |
| 48 | ACM2 | TEMP | MOD | 0 | 11 | 0 | 9 | 0 | 1 | 1 | 1 |
| 24 | ALL | | | 1306 | 266 | 970 | 731 | 265 | 17 | 91 | 21 |

ERROR PROCESSING STATISTICS...

| NUMBER | SEVERITY LEVEL |
|---|---|
| 0 | ERRORS |
| 0 | FATAL ERROR |

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

END. (Cont.)

Output Description

The Module Descriptor Block (MDB) summary shown in the sample output, reading left to right, gives the following statistics:

1. Module number on the file
2. Module name
3. Type of module
4. Mode of module (if function)
5. Language dialect of module
6. Parent module (if nested)
7. Statements in module
8. Executable statements in module
9. First executable statement of module
10. Number of parameters in module calling sequence
11. Number of entry points of module
12. Number of symbols in module
13. Number of DD-paths in module
14. Number of external invocations in module
15. Long name (if applicable)

The library header report lists mode of access (new or read), size of the library and number of modules, entries, fragments, and tokens.

The Interface Use Statistics shown in the sample output includes the number and name of the library, the type of access (permanent or temporary), the scope of the library (system or module), and miscellaneous access information.

## Description

This iteration command concludes a block of commands which are repeated for each specified module. There are two sequences of commands which select a number of modules and iterate a block of commands (which cannot contain another iteration). The two forms of command iteration are:

(1) FOR MODULE = (<$name_1$>, ..., <$name_n$>).
(commands)
END FOR.

(2) FOR ALL MODULES.
(commands)
END FOR.

## Rule

Maximum of 100 modules selected in command iteration loop.

Description

This command is used to reassign files. At installations where file numbers are used instead of names, the user should substitute the appropriate numbers. Appendix D has a chart with default assignments for each installation. The two forms of this command are:

FILE,LOG=<file-name>.
FILE,PUNCH=<file-name>.

The use of this command is optional. When it is present, the wrapup reports generated by the END command will be written on the output file; otherwise, these reports are not printed.

Rule

The only command that may precede this is SEGM.

AD-A065 447 GENERAL RESEARCH CORP SANTA BARBARA CALIF F/G 9/2

FORTRAN AUTOMATED VERIFICATION SYSTEM (FAVS). VOLUME II. USER'S--ETC(U)

JAN 79 D M ANDREWS, R A MELTON F30602-76-C-0436

UNCLASSIFIED RADC-TR-78-268-VOL-2 NL

2 OF 2

AD A065447

END

DATE FILMED

4--79

DDC

## Description

The following sequence selects each known module on the library and iterates a block of commands (which cannot contain another iteration) once for each module:

```
FOR ALL MODULES.
(commands)
END FOR.
```

## Rule

Maximum of 100 modules selected for this iteration command.

## Description

The action performed by the INSTRUMENT command is to write probed text statements to the PUNCH file. The INSTRUMENT command produces a small report as shown in the sample output.

## Rule

- A maximum of five testbounds may be specified.
- No FORTRAN labels between a range of 7777 and 8999.
- No routines named SPROB1 and SPROB2.
- The maximum DD-path number is 9999.

## Sample Output

```
DD-PATH INSTRUMENTATION OF MODULE BEGUN
CURRENT CPU TIME =       .194
DD-PATH INSTRUMENTATION OF MODULE COMPLETED
```

FOR MODULE = (<name$_1$>...

## Description

The following sequence selects a number of modules, by name, and iterates a block of commands (which cannot contain another iteration) once for each specified module:

FOR MODULE = (<name$_1$>, ..., <name$_n$>).<br>
(commands)<br>
END FOR.

## Rule

A maximum of 25 modules can be specified.

 INSTRUMENT,PUNCH,PROBE.

## Description

This command causes the data collection routine, SUBROUTINE SPROB2, to be written on the file LPUNCH. This routine is called by the software probes that are inserted into the user's program by the OPTION=INSTRUMENT. command or by the standard command, INSTRUMENT. During execution, the module name and probe number are recorded on the trace file LTEST each time this subroutine is invoked.

## Rule

This command must be used with the INSTRUMENT standard command to have the data collection routine written on LPUNCH.

## Sample Output

```
SUBROUTINE SPROB2        76/76    OPT=1                          FTN 4.6+433

      SUBROUTINE SPROB2(MODULE,ISTMT,IEXP)
      INTEGER MODULE(2)
      DATA LIEST/12/
      IDUM = IEXP
      GOTO 1
      ENTRY SPROB1
      IDUM = 0
    1 CONTINUE
      WRITE(LTEST) MODULE,ISTMT,IDUM
      RETURN
      END
```

 INSTRUMENT,IOPROBE=ON.

Description

This command causes FORTRAN code to be generated for INPUT and OUTPUT statements so that the values of variables listed in these statements will be printed out in the proper format.

Any variable whose type is not listed will not be printed. The syntax to provide type information is:

```
INPUT (/<type>/<variable list>,/<type>/<variable list>,...)
OUTPUT (/<type>/<variable list>,/<type>/<variable list>,...)
```

<type> may be REAL, INTEGER, HOLLERITH, or LOGICAL or the respective abbreviations for each, R , I , H , or L . <variable list> may contain non-scripted variable names, array names, individual elements of an array, or an array subrange. The INPUT and OUTPUT statements are turned into comments by FAVS, so they may be left in the code when the instrumented code will be compiled. See Sec. 5.6 for further details.

Rule

The type (REAL, INTEGER, LOGICAL, or Hollerith) must be specified for each variable listed in an INPUT or OUTPUT statement.

 INSTRUMENT,TESTBOUND<specs>.

## Description

This command is used to identify the end of one test execution test case and the beginning of another. It often is desirable to obtain the coverage analysis results for different parts of the instrumented source code. For example, the coverage within a single module might be of interest in addition to the coverage over an entire set of modules. This INSTRUMENT,TESTBOUND command could be used to define the beginning of a new test case at the beginning of the routine and the end of the test case at the end of the routine. When the instrumented code is executed, the coverage within this routine would appear as a separate test case; and, in addition, it would be included in the overall coverage results.

The two types of specifications which must be supplied with this command are

| | |
|---|---|
| MODULE = (<name>) | The name of the module in which the test boundary is to be located. |
| STATEMENT = <number> | The FAVS number of the statement at which the test boundary is to be located. |

INST,TESTBOUND,MODULE = (<name>), STATEMENT = <number>.

## Rules

1. A maximum of 5 testbounds may be specified.
2. This command must precede the INSTRUMENT command.

Description

This command identifies the language of the source code to be analyzed by FAVS. The two possibilities for <name> are as follows:

- DMATRAN
- FORTRAN (default)

Rule

This command (if supplied) must precede the START command or any OPTION command. It is not required with OPTION = RESTRUCTURE.

MODULE = (<name>).

**Description**

Modules are known to FAVS by their names. The following identifies a specific module as the one to which subsequent commands apply:

MODULE = (<name>).

**Rule**

If there are duplicate module names in the library the module selector will choose the last one.

Description

This command specifies that a new called <name> is to be created by the current FAVS run, where <name> is any four character word. This command is implicitly generated automatically if an OLD LIBRARY command is not supplied. In this case, the <name> generated is blank.

Rule

If this command is used, it must precede the START command or the OPTION selection macro command.

## Description

This command specifies that an old library is to be used during the current run. The name identifying the library need not be the same as when the library was created.

## Rule

If this command is used, it must precede the START command or the OPTION selection macro command.

## Description

This command produces a detailed listing of the source statements on each DD-path for the current module. DD-path descriptions are also included. The report is similar in format to the report from the PRINT, MODULE command.

## Sample Output

```
DD-PATH DEFINITIONS              SUBROUTINE EXAMPL ( INFO, LENGTH )

------------------------------------------------------------------------------------------------
 1                   SUBROUTINE EXAMPL ( INFO, LENGTH )
                                                                ** DDPATH   1 IS PROCEDURE ENTRY
 2           C
 3           C       ILLUSTRATION OF DMATRAN SYNTAX
 4           C
 5                   IF ( INFO .LE. 10 .AND. LENGTH .GT. 0 ) THEN
                                                                ** DDPATH   2 IS TRUE BRANCH
                                                                ** DDPATH   3 IS FALSE BRANCH
 6 ( 1)              . CALL CALLER ( INFO )
 7                   ELSE
 8 ( 1)              . LENGTH = 50
 9                   ENDIF
10                   CASEOF ( INFO + 6 )
                                                                ** DDPATH   4 IS BRANCH OUTWAY  1
                                                                ** DDPATH   5 IS BRANCH OUTWAY  2
                                                                ** DDPATH   6 IS BRANCH OUTWAY  3
11                   CASE ( 14 )
12 ( 1)              . LENGTH = LENGTH - INFO
13                   CASE ( 17 )
14 ( 1)              . DOWHILE ( INFO .LT. 20 )
                                                                ** DDPATH   7 IS LOOP AGAIN
                                                                ** DDPATH   8 IS LOOP ESCAPE
15 ( 2)              . . DOUNTIL ( LENGTH .LE. INFO )
16 ( 3)              . . . INVOKE ( COMPUTE LENGTH )
17 ( 3)              . . . IF ( LENGTH .GE. 30 ) THEN
                                                                ** DDPATH   9 IS TRUE BRANCH
                                                                ** DDPATH  10 IS FALSE BRANCH
18 ( 4)              . . . . INVOKE ( PRINT-RESULTS )
19 ( 3)              . . . ENDIF
20 ( 2)              . . ENDUNTIL
                                                                ** DDPATH  11 IS LOOP ESCAPE
                                                                ** DDPATH  12 IS LOOP AGAIN
21 ( 2)              . . INFO = INFO + 1
22 ( 1)              . ENDWHILE
23                   CASEELSE
24 ( 1)              . DOWHILE ( LENGTH .GT. 0 )
                                                                ** DDPATH  13 IS LOOP AGAIN
                                                                ** DDPATH  14 IS LOOP ESCAPE
25 ( 2)              . . INVOKE ( COMPUTE LENGTH )
26 ( 1)              . ENDWHILE
27                   ENDCASE
28                   BLOCK ( PRINT-RESULTS )
                                                                ** DDPATH  15 IS A PROCEDURE ENTRY
29 ( 1)              . WRITE ( 6, 1 ) INFO, LENGTH
30 ( 1)        1     . FORMAT (10X,I5,20X,I5)
31                   ENDBLOCK
32                   BLOCK ( COMPUTE LENGTH )
                                                                ** DDPATH  16 IS A PROCEDURE ENTRY
33 ( 1)              . LENGTH = LENGTH - 10
34                   ENDBLOCK
35                   RETURN
36                   END
------------------------------------------------------------------------------------------------
```

## Description

This command produces a detailed listing of the source text of the current module. The salient features are:

- Sequential statement numbers assigned by FAVS
- A number for the level of nesting of the statement
- Indentation of statements to show nesting levels
- Inclusion of DD-path numbers if STRUCTURAL analysis is performed

## Rule

The BASIC command must be used with this command to obtain a module listing. The DD-path numbers will not be included unless the STRUCTURAL command is also used.

## Sample Output

```
STATEMENT LISTING                     SUBROUTINE EXAMPL ( INFO, LENGTH )

 NO. LEVEL   LABEL  STATEMENT TEXT...                                        DDPATHS
------------------------------------------------------------------------------------
  1                 SUBROUTINE EXAMPL ( INFO, LENGTH )                      (  1)
  2           C
  3           C     ILLUSTRATION OF DMATRAN SYNTAX
  4           C
  5                 IF ( INFO .LE. 10 .AND. LENGTH .GT. 0 ) THEN            (  2-  3)
  6 ( 1)            .  CALL CALLER ( INFO )
  7                 ELSE
  8 ( 1)            .  LENGTH = 50
  9                 ENDIF
 10                 CASEOF ( INFO + 6 )                                     (  4-  6)
 11                 CASE ( 14 )
 12 ( 1)            .  LENGTH = LENGTH - INFO
 13                 CASE ( 17 )
 14 ( 1)            .  DOWHILE ( INFO .LT. 20 )                             (  7-  8)
 15 ( 2)            .  .  DOUNTIL ( LENGTH .LE. INFO )
 16 ( 3)            .  .  .  INVOKE ( COMPUTE LENGTH )
 17 ( 3)            .  .  .  IF ( LENGTH .GE. 30 ) THEN                     (  9- 10)
 18 ( 4)            .  .  .  .  INVOKE ( PRINT-RESULTS )
 19 ( 3)            .  .  .  ENDIF
 20 ( 2)            .  .  ENDUNTIL                                          ( 11- 12)
 21 ( 2)            .  .  INFO = INFO + 1
 22 ( 1)            .  ENDWHILE
 23                 CASEELSE
 24 ( 1)            .  DOWHILE ( LENGTH .GT. 0 )                            ( 13- 14)
 25 ( 2)            .  .  INVOKE ( COMPUTE LENGTH )
 26 ( 1)            .  ENDWHILE
 27                 ENDCASE
 28                 BLOCK ( PRINT-RESULTS )                                 ( 15)
 29 ( 1)            .  WRITE ( 6, 1 ) INFO, LENGTH
 30 ( 1)      1     .  FORMAT (10X,I5,20X,I5)
 31                 ENDBLOCK
 32                 BLOCK ( COMPUTE LENGTH )                                ( 16)
 33 ( 1)            .  LENGTH = LENGTH - 10
 34                 ENDBLOCK
 35                 RETURN
 36                 END
------------------------------------------------------------------------------------
```

## Description

This report classifies each statement of a module as either a declaration, executable, decision, or documentation statement. Under these classifications, a tabulation of the subtypes is listed.

## Sample Output

```
STATEMENT PROFILE                    SUBROUTINE EXAMPL ( INFO, LENGTH )

  INTERFACE CHARACTERISTICS
  -------------------------

                 ARGUMENTS           2
                 ENTRY               1
                 EXIT                1
                 INTERNAL PROCEDURES 2
                 INVOKES             4
                 WRITE               1

  STATEMENT      STATEMENT
  CLASSIFICATION TYPE            NUMBER          PERCENT
-------------------------------------------------------------------------------------------

  DECLARATION...

                 FORMAT               1              2.8
                 -----------------------------------------
                 TOTAL                1              2.8

  EXECUTABLE...

                 ASSIGNMENT           4             11.1
                 CALL                 1              2.8
                 CASE                 2              5.6
                 CASEELSE             1              2.8
                 DOUNTIL              1              2.8
                 ELSE                 1              2.8
                 ENDBLOCK             2              5.6
                 ENDCASE              1              2.8
                 ENDIF                2              5.6
                 ENDWHILE             2              5.6
                 END                  1              2.8
                 INVOKE               3              8.3
                 RETURN               1              2.8
                 WRITE                1              2.8
                 -----------------------------------------
                 TOTAL               23             63.9

  DECISION...

                 BLOCK                2              5.6
                 CASEOF               1              2.8
                 DOWHILE              2              5.6
                 ENDUNTIL             1              2.8
                 IFTRAN-IF            2              5.6
                 SUBROUTINE           1              2.8
                 -----------------------------------------
                 TOTAL                9             25.0

  DOCUMENTATION...

                 COMMENT              3              8.3
                 -----------------------------------------
                 TOTAL                3              8.3

-------------------------------------------------------------------------------------------
 * TOTAL PERCENTAGE MAY BE MORE THAN 100 BECAUSE OF OVERLAPPING CLASSIFICATIONS
```

## Description

The SEGMENT command causes the list of SEGMENT commands, generated from the user commands, to be output.

## Rule

This must be the very first FAVS command.

## Sample Output

```
SEGM.
OPTI=LIST,STAT,SUMM,DOCU,INST,INPU,REAC.
TESTBOUND,MODULE=(CLASS), STATEMENT=50.
REACHING SET,MODULE=(CLASS), TO=45,FROM=40.

THE ABOVE MACRO COMMANDS EXPAND TO THE FOLLOWING SERIES
STAR.
BASI.
FORA.
BUIL,DMT .
BUIL,PRED.
ENDF.
BUIL,PARM.
BUIL,CROS.
FORA.
STRU.
ENDF.
FORA.
PRIN,DDPA.
STAT.
DOCU,INVO.
DOCU,BAND.
PRIN,PROF.
ENDF.
DOCU,MATR,LIBR.
DOCU,COMM,PRIN=SUMM.
DOCU,READ.
DOCU,CROS.
INST,TESTBOUND,MODULE=(CLASS),STATEMENT=50.
MODULE=(CLASS)
ASSI,REACHING SET,TO=45,FROM=40
INST,PUNC,PROB.
FORA.
INST,IOPR=ON  .
INST.
ENDF.
END .
```

## Description

The START command terminates the LANGUAGE and LIBRARY description commands and signals the beginning of processing.

## Rule

The LANGUAGE, FILENAME, and LIBRARY commands (if used) must precede the START command. All other commands must follow it.

## Sample Output

```
SYSTEM STARTUP...

FILE DEFINITIONS...

LIN      = INPUT FILE.
OUTPUT   = OUTPUT FILE.
OUTPUT   = SYSTEM LOG FILE.
OUTPUT   = ERROR REPORT FILE.
OUTPUT   = DEBUGGING MESSAGE FILE.
LIBNEW   = PERMANENT LIBRARY FILE.
LIBWSP   = TEMPORARY LIBRARY FILE.
LPUNCH   = PUNCH FILE.
LTEST    = EXECUTION TRACE FILE.
LSOURC   = TEMPORARY SOURCE FILE.
LTEMP    = TEMPORARY SOURCE FILE.
STAPE    = SOURCE TAPE FILE.

NO MODULES CURRENTLY IN SYSTEM

LIBRARY HEADER ---

    MODE OF ACCESS              NEW
    FRAGMENT SIZE               500
    LIBRARY SIZE                  0
    NUMBER OF MODULES             0
    NUMBER OF ENTRIES             0
    NUMBER OF FRAGMENTS           0
    NUMBER OF TOKENS            234
```

## Description

The actions performed by the STRUCTURAL command are:

1. To build tables describing the graphical characteristics of the specified module and to add them to a library.

2. To produce the report shown in the sample output.

The output is written on the LOG file and may be obtained by using the command, FILENAME, LOG = OUTPUT.

## Rule

- The maximum number of DD-paths which can begin at a statement is 50.
- The maximum number of statements on a single DD-path is 100.

## Sample Output

```
STRUCTURAL ANALYSIS CF MODULE <EXAMPL  > BEGUN
CURRENT CPU TIME =      5.375
NUMBER OF DDPATHS FOUND =      1
STRUCTURAL ANALYSIS OF MODULE COMPLETED
```

 STRUCTURAL,COMPUTE=FULL.

Description

This command should only be used with the RESTRUCTURE option for generating DMATRAN modules from FORTRAN code. It indicates to FAVS to perform additional structural analyses.

Rule

This command must be used with, and only with, the RESTRUCTURE command.

## Description

This command should only be used with the RESTRUCTURE option for generating DMATRAN modules from FORTRAN code. It indicates to FAVS to perform additional structural analyses.

## Rule

This command must be used with, and only with, the RESTRUCTURE command.

## APPENDIX C

## COMMAND SUMMARY AND CHECKLIST

## FAVS COMMANDS

The option selection command is required; the other commands are used when it is appropriate, and they must appear in the order shown. Where an abbreviation is allowed, it appears to the right of the command.

| | |
|---|---|
| RESTART or EXPAND | REST. or EXPA. |

Instructs FAVS to use a saved restart file from a previous run. EXPAND allows additional source to be added to a restart file.

| | |
|---|---|
| LANGUAGE=DMATRAN. | LANG = DMAT. |

The default is FORTRAN, in which case the command is not required.

| | |
|---|---|
| FILE,PUNCH=<file-name>. | FILE,PUNC=<file-name>. |

Instructs FAVS to reassign the punch file.

| | |
|---|---|
| OPTIONS=<list> | OPTI = <list> |

<list> may contain one or more of the following options, separated by commas:

| | |
|---|---|
| LIST | LIST |
| DOCUMENT | DOCU |
| SUMMARY | SUMM |
| STATIC | STAT |
| INSTRUMENT | INST |
| INPUT/OUTPUT | INPU |
| REACHING SET | REAC |
| RESTRUCTURE | REST |

FOR MODULE = (<name1>,<name2>,....)

module selection command.

TESTBOUND,MODULE = (<name>),STATEMENT = <number>

Used with instrumentation command for setting test case boundaries.

REACHING SET,MODULE = (<name>),TO = <DD-path number>,
FROM = <DD-path number>,{ITERATIVE}

When the option, REACHING SET, is used, it is necessary to specify one or more reaching sets with the above command. The use of ITERATIVE is optional; if present, an iterative reaching set is generated.

## FAVS CHECKLIST

ALL OPTIONS

When using any FAVS option, compile source code to be certain it is free of syntax errors. If the text is to be instrumented for any of the dynamic tests, it must have executed properly.

INSTRUMENT

Perform an execution test on the program before submitting it to FAVS for instrumentation. When a main program is not being instrumented, a "test end" must be specified within the set of modules that are being instrumented. The number of the exit statement in the last module which will be executed should be supplied with the command,

TESTBOUND,MODULE = (<name>),STATEMENT = <number>

INPUT/OUTPUT

Add INPUT and OUTPUT statements to the routines where a report on the values of the variables is desired.

RESTRUCTURE

The command set should contain this option alone, since no others will be processed at the same time.

REACHING SET

No reaching set processing will take place unless there is at least one reaching set specified. The form is:

REACHING SET,MODULE = (<name>),TO = <DD-path number>
FROM = <DD-path number>,{ITERATIVE}.

Section 5 contains universal and syntax constraints as well as individual option constraints.

## ANALYZER COMMANDS

Selection of ANALYZER reports desired must be made by the user. The type of report is specified in the command,

OPTION(S) = <list>

<list> may contain one or more of the following options, separated by commas:

| | |
|---|---|
| DETAILED | DETA |
| NOTHIT | NOTH |
| SUMMARY | SUMM |

When the DETAILED option is listed, reports will be generated only for those modules that are listed in a command,

FOR MODULE(S) = ($<name_1>$,$<name_2>$, ...,$<name_n>$).

<name> is the name of the subroutine, function or program. This module selection command must precede the OPTION = DETAILED command.

# APPENDIX D

# FILE DESCRIPTIONS

TABLE D.1

FILES USED IN FAVS PROCESSING AT RADC INSTALLATION

| FILE NUMBER | FILE NAME | DATA STRUCTURE | MODE (1) | STORAGE FORM(2) | RECORD FORMAT | RECOMMENDED ALLOCATION | USAGE |
|---|---|---|---|---|---|---|---|
| 1 | LIBNEW | library | B | R | system standard (4) | permanent file or scratch file | R/W |
| 2 | LIBWSP | workspace | B | R | system standard (4) | scratch file | R/W |
| 4 | LCOMMD | user commands | H | S | card image | scratch file | R/W |
| 5 | LCOMIN | commands input | H | S | card image | system card reader permanent file | R |
| 6 | LOUT | reports | H | S | 128 characters/ line maximum | system printer | W |
| 7 | LPUNCH | instrumented/ restructured source | H | S | card image | system punch or permanent file | W |
| 8 | LSOURCE | temporary source file | H | S | card image | scratch file | R/W |
| 9 | LIN | source | H | S | card image | system card reader permanent file | R |
| 10 | LTEMP | temporary source file | H | S | card image | scratch file | R/W |
| 12 | LTEST | probe test data trace file | B | S | card image | system card reader permanent file | R |

Notes: (1) B = binary; H = character
(2) R = random; S = sequential
(3) R = read only; R/W = read and/or write; W = write only
(4) Installation dependent

TABLE D.2

FILES USED IN FAVS PROCESSING AT DMA INSTALLATIONS

| FILE NUMBER | FILE NAME | DATA STRUCTURE | MODE (1) | STORAGE FORM(2) | RECORD FORMAT | RECOMMENDED ALLOCATION | USAGE |
|---|---|---|---|---|---|---|---|
| 1 | LIBNEW | library | B | R | system standard (4) | permanent file or scratch file | R/W |
| 2 | LIBWSP | workspace | B | R | system standard (4) | scratch file | R/W |
| 4 | LCOMMD | user commands | H | S | card image | scratch file | R/W |
| 5 | LCOMIN | commands input | H | S | card image | system card reader permanent file | R |
| 6 | LOUT | reports | H | S | 128 characters/ line maximum | system printer | W |
| 9 | LPUNCH | instrumented/ restructured source | H | S | card image | system punch or permanent file | W |
| 8 | LSOURCE | temporary source file | H | S | card image | scratch file | R/W |
| 5 | LIN | source | H | S | card image | system card reader permanent file | R |
| 10 | LTEMP | temporary source file | H | S | card image | scratch file | R/W |
| 12 | LTEST | probe test data trace file | B | S | card image | system card reader permanent file | R |

Notes:
(1) B = binary; H = character
(2) R = random; S = sequential
(3) R = read only; R/W = read and/or write; W = write only
(4) Installation dependent

- FAVS INITIAL RUN - CREATES A RESTART FILE

```
@HDG                             **  FAVS INITIAL RUN **
@ASG,A  YOURSOURCE.              . YOUR FORTRAN OR DMATRAN SOURCE
@USE    Y.,YOURSOURCE.           .
@ASG,CP YOURFILE,F40///400       . (OPTIONAL) CATALOG FAVS RESTART FILE
@ASG,A  DBM*FAVS-DMA.            . ASG FAVS, TRAN, ANALYZER, TEMPFILES
@USE    R.,DBM*FAVS-DMA.         .
@ADD,P  R.TEMPFILES              . ASG TEMPORARY FILES
@XQT    R.FAVS                   . EXECUTE FAVS
LANGUAGE=DMATRAN.                . (OPTIONAL)
OPTION=-------.                  . ANY LIST OF VALID OPTIONS (SEC 3.)
FOR MODULES=(LIST OF MODULES).   . (OPTIONAL) DEFAULT IS ALL MODULES
@EOF                             . SEPARATES FAVS COMMANDS FROM YOUR SOURCE
@ADD,P  Y.PROCS                  . (OPTIONAL) ADD PROCS HERE
@ADD,P  Y.ELEMENTS               . ADD SOURCE ELEMENTS HERE
@COPY   2.,YOURFILE              . (OPTIONAL) SAVE RESTART FILE
@FIN
```

- FAVS RESTART RUN - USES A RESTART FILE

```
@HDG                             **  FAVS RESTART RUN **
@ASG,A  YOURFILE.                . ASG FAVS RESTART FILE (FROM PREVIOUS RUN)
@ASG,A  DBM*FAVS-DMA.            . ASG FAVS, TRAN, ANALYZER, TEMPFILES
@USE    R.,DBM*FAVS-DMA.         .
@ADD,P  R.TEMPFILES              . ASG TEMPORARY FILES
@COPY   YOURFILE.,2.             . MAKE TEMPORARY RESTART FILE
@XQT    R.FAVS                   . EXECUTE FAVS
RESTART                          . USE RESTART FILE
LANGUAGE=DMATRAN.                . (OPTIONAL)
OPTION=-------.                  . ANY LIST OF VALID OPTIONS (SEC 3.)
FOR MODULES=(LIST OF MODULES).   . (OPTIONAL) DEFAULT IS ALL MODULES
@FIN
```

- FAVS INSTRUMENT, EXECUTE, AND ANALYZE RUN

```
@HDG                         **  FAVS INSTRUMENT, EXECUTE, AND ANALYZE RUN **
@ASG,A  YOURSOURCE.           .  YOUR FORTRAN OR DMATRAN SOURCE
@USE    Y.,YOURSOURCE.        .
@ASG,CP YOURFILE,F40///400    .  (OPTIONAL) CATALOG FAVS RESTART FILE
@ASG,A  DBM*FAVS-DMA.         .  ASG FAVS, TRAN, ANALYZER, TEMPFILES
@USE    R.,DBM*FAVS-DMA.      .
@ADD,P  R.TEMPFILES           .  ASG TEMPORARY FILES
@XQT    R.FAVS                .  EXECUTE FAVS
LANGUAGE=DMATRAN.             .  (OPTIONAL)
OPTION=INSTRUMENT,-----.      .  ANY LIST OF VALID OPTIONS (SEC 3.)
FOR MODULES=(LIST OF MODULES). . (OPTIONAL) DEFAULT IS ALL MODULES
@EOF                          .  SEPARATES FAVS COMMANDS FROM YOUR SOURCE
@ADD,P  Y.PROCS               .  (OPTIONAL) ADD PROCS HERE
@ADD,P  Y.ELEMENTS            .  ADD SOURCE ELEMENTS HERE
@COPY   2.,YOURFILE           .  (OPTIONAL) SAVE RESTART FILE
@XQT R.TRAN                   .  (OPTIONAL) REQUIRED WHEN LANGAUGE=DMATRAN
@ADD,P 9.                     .  YOUR INSTRUMENTED SOURCE IS ON 9.
@MAP                          .  MAP FOR YOUR PROGRAM
@XQT                          .  EXECUTE YOUR INSTRUMENTED PROGRAM
        ( YOUR DATA )
@XQT    R.ANALYZER            .  EXECUTE COVERAGE ANALYZER
FOR MODULES=( LIST OF INSTRUMENTED ELEMENTS).
OPTION=-------.               .  ANY LIST OF VALID OPTIONS (SEC. 6)
@FIN
```

- FAVS RESTRUCTURE RUN

```
@HDG                         **  FAVS RESTRUCTURE RUN **
@ASG,A  YOURSOURCE.           .  YOUR FORTRAN SOURCE
@USE    Y.,YOURSOURCE.        .
@ASG,CP YOURFILE,F40///400    .  (OPTIONAL) CATALOG FAVS RESTART FILE
@ASG,A  DBM*FAVS-DMA.         .  ASG FAVS. TRAN, ANALYZER, TEMPFILES
@USE    R.,DBM*FAVS-DMA.      .
@ADD,P  R.TEMPFILES           .  ASG TEMPORARY FILES
@XQT    R.FAVS                .  EXECUTE FAVS
OPTION=RESTRUCTURE.           .
@EOF                          .  SEPARATES FAVS COMMANDS FROM YOUR SOURCE
@ADD,P  Y.PROCS               .  (OPTIONAL) ADD PROCS HERE
@ADD,P  Y.ELEMENTS            .  ADD SOURCE ELEMENTS HERE
@COPY   2.,YOURFILE           .  (OPTIONAL) SAVE RESTART FILE
@XQT    R.TRAN                .  EXECUTE DMATRAN PRECOMPILER
@ADD,P 9.                     .  RESTRUCTURED SOURCE IS ON 9.
```

- FAVS EXPAND RUN - EXPANDS A RESTART FILE

```
@HDG                          **     FAVS EXPAND RUN **
@ASG,A   YOURSOURCE.               . YOUR FORTRAN OR DMATRAN SOURCE
@USE     Y.,YOURSOURCE.            .
@ASG,A   YOURFILE.                 . ASG FAVS RESTART FILE (FROM PREVIOUS RUN)
@ASG,A   DBM*FAVS-DMA.             . ASG FAVS, TRAN, ANALYZER, TEMPFILES
@USE     R.,DBM*FAVS-DMA.          .
@ADD,P   R.TEMPFILES               . ASG TEMPORARY FILES
@COPY    YOURFILE.,2.              . MAKE TEMPORARY RESTART FILE
@XQT     R.FAVS                    . EXECUTE FAVS
EXPAND.                            . EXPAND RESTART FILE WITH NEW SOURCE ELEMENTS
LANGUAGE=DMATRAN.                  . (OPTIONAL)
OPTION=-------.                    . ANY LIST OF VALID OPTIONS (SEC 3.)
FOR MODULES=(LIST OF MODULES).     . (OPTIONAL) DEFAULT IS ALL MODULES
@EOF                               . SEPARATES FAVS COMMANDS FROM YOUR SOURCE
@ADD,P   Y.PROCS                   . (OPTIONAL) ADD PROCS HERE
@ADD,P   Y.ELEMENTS                . ADD SOURCE ELEMENTS HERE
@COPY    2.,YOURFILE               . (OPTIONAL) SAVE EXPANDED RESTART FILE
@FIN
```

- FAVS STATIC RUN

```
@HDG                          **     FAVS STATIC RUN **
@ASG,A   YOURSOURCE.               . YOUR FORTRAN OR DMATRAN SOURCE
@USE     Y.,YOURSOURCE.            .
@ASG,A   DBM*FAVS-STUBS.           . ASG RESTART FILE DESCRIBING FORTRAN SYSLIB
@ASG,A   DBM*FAVS-DMA.             . ASG FAVS, TRAN, ANALYZER, TEMPFILES
@USE     R.,DBM*FAVS-DMA.          .
@ADD,P   R.TEPFILES                . ASG TEMPORARY FILES
@COPY    DBM*FAVS-STUBS.,2.        . MAKE TEMPORARY RESTART FILE
@XQT     R.FAVS                    . EXECUTE FAVS
EXPAND                             . EXPAND RESTART FILE WITH NEW SOURCE ELEMENTS
LANGUAGE=DMATRAN.                  . (OPTIONAL)
OPTION=STATIC,------.              . ANY LIST OF VALID OPTIONS (SEC 3.)
FOR MODULES=(LIST OF MODULES).     . (OPTIONAL) DEFAULT IS ALL MODULES
@EOF                               . SEPARATES FAVS COMMANDS FROM YOUR SOURCE
@ADD,P   Y.PROCS                   . (OPTIONAL) ADD PROCS HERE
@ADD,P   Y.ELEMENTS                  ADD SOURCE ELEMENTS HERE
@FIN
```

## RADC HONEYWELL 6180/MULTICS
## (USING THE GCOS ENCAPSULATOR)

The job stream in Fig. E.1 can be used for executing any of the FAVS options: LIST, SUMMARY, DOCUMENT, STATIC, INSTRUMENT, INPUT/OUTPUT, REACHING SET.

```
1.    $     snumb
2.    $     ident
3.    $     program rlhs
4.    $     limits  (CP time limit),52k,,(print line limit)
5.    $     prmfl   h*,r,r,>udd>3201c0320>Urban>favs>hstar
6.    $     select  >udd>3201c0320>Urban>favs>filedefs -ascii
7.    $     prmfl   09,r,s,>udd>(BCD source code)
8.    $     prmfl   07,w,s,>udd>(BCD instrumented source code)
9.    option=inst,list,summ,docu,stat,inpu.
10.   $     endjob
```

Figure E.1. Sample FAVS Job Stream

Notes

1. If a large amount of source code is to be analyzed, insert the following file card to increase the random data base file size from 10R:

   ```
   $     file     01,z2r,(size in links)r
   ```

2. The BCD source code (control card 7) must be standard, card-image FORTRAN (not MULTICS FORTRAN).

3. If no instrumentation or restructuring is to be performed, delete control card 8.

4. The above job stream can also be used for restructuring, using the FAVS command

   option = rest.

   File 07 will contain the output DMATRAN source code.

5. If the data base library is to be saved for a subsequent "restart" run (e.g., for obtaining reaching set information), the following permanent file card must be inserted after control card 6, 7 or 8:

```
$    prmfl    01,r/w,r,>udd>(data base file)
```

When the restart activity is to be performed, precede the FAVS option command with:

```
restart
```

and insert the above permanent file card after control card 6, 7 or 8.

The instrumented source file contains the FAVS data collection routine. The instrumented source code should be compiled and executed in the same manner as uninstrumented source code with the following exception: the trace file (file code 12) must be available during execution. If the execution coverage analysis is to be performed in a separate job, the trace file must be saved on a magnetic tape or on permanent disk space. If the coverage analysis is an additional activity of the execution job, a temporary file can be used.

```
e.g.,    $    file     12,x2s,(size in links)1
or       $    prmfl    12,w,s,>udd>(trace file)
```

The job stream in Fig. E.2 can be used for obtaining execution coverage analysis if the trace data file was saved on permanent disk space during execution of the instrumented code.

```
1.   $   snumb   (number)
2.   $   ident
3.   $   program rlhs
4.   $   limits  (CP time limit),30k,,(print line limit)
5.   $   prmfl   H*,r,r,>udd>3201c0320>Urban>analyzer>hstar
6.   $   prmfl   12,r,s,>udd>(execution trace file)
7.   for modules = (name1,...,namen)
8.   option = summary, nothit, detailed
9.   $   endjob
```

Figure E.2. Sample Coverage Analysis Job Stream

---

In the event that unusually large programs are to be processed, it may be required, due to resource limitations, to build a data base on permanent storage media by running several successive executions, each operating on a separate file of modules. The job stream in Fig. E.3, utilizing FAVS segment commands, can be used for this purpose.

```
1.   $   snumb   (number)
2.   $   ident
3.   $   program rlhs
4.   $   limits  (CP time limit),52k,,(print line limit)
5.   $   prmfl   h*,r,r,>udd>3201c0320>Urban>favs>hstar
6.   $   select  >udd>3201c0320>Urban>favs>filedefs -ascii
7.   $   prmfl   09,r,s,>udd>(BCD source code)
8.   $   prmfl   01,r/w,r,>udd>(data base file)
9.   EXPAND.
10.  OPTION=LIST.
11.  $   endjob
```

Figure E.3. Incremental Data Base Creation

---

When the data base is completed (all BCD source code files have been processed), it may be utilized for normal FAVS processing. For each subsequent FAVS job, the "RESTART" capability must be used. The BCD source file (09) is no longer required.

## RADC HONEYWELL 6180/GCOS

The job stream in Fig. E.4 can be used for executing any of the following FAVS options: LIST, SUMMARY, DOCUMENT, STATIC, INSTRUMENT, INPUT/OUTPUT, REACHING SET.

```
1.   $    IDENT
2.   $    SELECT  BFCBGRC4/FAVS/EXECUTE
3.   $    PRMFL   09,R,S,(SOURCE)
4.   $    PRMFL   07,W,S,(INSTRUMENTED SOURCE)
     [FAVS COMMANDS/OPTIONS]
5.   $    ENDJOB
```

Figure E.4. Sample FAVS Job Stream

---

**NOTES**

1. If a large amount of source code is to be analyzed, insert the following file card to increase the random data base file size from 10R:

   $ FILE 01,Z2R,(SIZE IN LINKS)R

2. The BCD source code (File 09) must be standard card-image FORTRAN.

3. If no instrumentation is to be performed, delete File 07.

4. If restructuring is to be performed, use the FAVS command OPTION = RESTRUCTURE. It must be performed independent of any other option. File 07 will contain the output structured source code.

5. If the data base library is to be saved for a subsequent "RESTART" run (e.g., for obtaining reaching set information, for instance), the following permanent file card must be inserted after the $ SELECT CARD:

   $ PRMFL 01,R/W,R,(DATA BASE FILE)

When the restart activity is to be performed, precede the FAVS option command with:

RESTART

Following instrumentation, the instrumented source file (File 07) contains the FAVS data collection routine in addition to the instrumented FORTRAN/DMATRAN source code. The instrumented source code should be compiled and executed in the same manner as uninstrumented source code with the following exception: the trace file (File 12) must be available during execution. (Also note that if the instrumented source code is DMATRAN, it must be processed by the DMATRAN pre-compiler before being compiled and executed (see DMATRAN User's Guide, General Research Corporation CR-1-673/1.

If the execution coverage analysis is to be performed in a separate job, the trace file must be saved on a magnetic tape or on permanent disk space. If the coverage analysis is an additional activity of the execution job, a temporary file can be used.

```
$    PRMFL    12,W,S,(TRACE FILE)
     or
$    FILE     12,X2S,(SIZE IN LINKS)L
```

The job stream in Fig. E.5 can be used for obtaining execution coverage analysis (OPTIONS SUMMARY, NOTHIT, DETAILED), if the trace file was saved on permanent disk space during execution of the instrumented code.

```
1.   $    IDENT
2    $    SELECT   BFCBGRC4/ANALYZER/EXECUTE
3.   $    PRMFL    12,R,S,(TRACE FILE)
               [FAVS ANALYZER COMMANDS/OPTIONS]
4.   $    ENDJOB
```

Figure E.5. Sample Coverage Analysis Job Stream

In the event that unusually large programs are to be processed, it may be required, due to resource limitations, to build a data base on permanent media by running several successive executions, each operating on a separate file of modules. The job stream in Fig. E.6, utilizing FAVS standard commands, can be used for this purpose.

```
1.    $      IDENT
2.    $      SELECT     BFCBGRC4/FAVS/EXECUTE
3.    $      PRMFL      01,R/W,R,(DATA BASE FILE)
4.    $      PRMFL      09,R,S,(BCD Source File)
5.    EXPAND.
6.    OPTION=LIST.
7.    $      ENDJOB
```

Figure E.6. Incremental Data Base Creation

---

When the data base is completed (all BCD source code files have been processed), it may be utilized for normal FAVS processing. For each subsequent FAVS job, the "RESTART" capability must be used. The BCD source file (09) is no longer required.

INDEX

INDEX (Cont'd.)

INDEX (Cont'd.)

## FAVS COMMANDS

[REST[ART].]

[EXPA[ND].]

[FILE,PUNC[H]=<file number>.]

[LANG[UAGE]=DMAT[RAN].]

{OPTI[ONS]= option {, option}.}

where option is one of:

DOCU[MENT]
INPU[T/OUTPUT]
INST[RUMENT]
LIST
REAC[HING SET]
REST[RUCTURE]
STAT[IC]
SUMM[ARY]

{FOR M[ODULES] = (<name> {, <name>}).}

{TEST[BOUND],MODU[LE]=(<name>),STAT[EMENT]=<number>.}

{REAC[HING SET],MODU[LE]=(name),
*TO=<DD-path number>* [, FROM=<DD-path number>]
[,ITER[ATIVE]].}

## ANALYZER COMMANDS

{OPTI[ONS]=option {, option}.}

where option is one of:

DETAILED
NOTHIT
SUMMARY

{FOR M[ODULES]=(<name> {, <name>}).}

[ ] = optional

{ } = optional an arbitrary number of times

< > = integer constant or character string

# MISSION
## of
## Rome Air Development Center

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications ($C^3$) activities, and in the $C^3$ areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

AMERICAN REVOLUTION BICENTENNIAL 1776-1976